**IN THE UNITED STATES DISTRICT COURT**
**FOR THE DISTRICT OF DELAWARE**

| | | |
|---|---|---|
| ORACLE CORPORATION and ORACLE U.S.A. INC., | ) | |
| Plaintiffs/Counterclaim Defendants, | ) | C.A. No. 06-414-SLR |
| v. | ) | **JURY TRIAL DEMANDED** |
| EPICREALM LICENSING, LP, | ) | **PUBLIC VERSION** |
| Defendant/Counterclaim Plaintiff. | ) | |

**APPENDIX IN SUPPORT OF EPICREALM'S MOTION FOR PARTIAL SUMMARY JUDGMENT OF LITERAL INFRINGEMENT**

**VOLUME I of III**

**EXHIBITS 1 to 13**

Richard L. Horwitz (#2246)
David E. Moore (#3983)
POTTER ANDERSON & CORROON LLP
Hercules Plaza, 6th Floor
1313 N. Market Street
Wilmington, DE 19899
Tel: (302) 984-6000
rhorwitz@potteranderson.com
dmoore@potteranderson.com

*Attorneys for Defendant/Counterclaim Plaintiff epicRealm Licensing, LP*

OF COUNSEL:

Harry J. Roper
George S. Bosy
Aaron A. Barlow
Patrick L. Patras
David R. Bennett
Paul D. Margolis
Benjamin J. Bradford
Emily C. Johnson
JENNER & BLOCK
330 N. Wabash Avenue
Chicago, IL 60611-7603
Tel: (312) 923-8305

Dated: July 31, 2008
Public Version Dated: August 7, 2008
876825 / 31393 / Oracle

## TABLE OF CONTENTS


| Exh. | Document |
|---|---|
| 1. | U.S. Patent 5,894,554, Lowery et al. |
| 2. | U.S. Patent 6,415,335, Lowery et al. |
| 3. | Excerpt from Report of Plaintiffs' Expert Michael I. Shamos Concerning Invalidity, May 2, 2008 |
| 4. | Excerpt from Oracle WebServer User's Guide, Release 1.0 A34986-2, ORCL00941713-00941960 |
| 5. | The Oracle WebServer: A Technical Discussion, ORCL01042968-01042974 |
| 6. | Oracle WebServer Release 2.0, Feature List, 10/30/95, ORCL00937853-00937862 |
| 7. | Excerpt from Expert Witness Rebuttal Report on Non-Infringement By Dr. Paul C. Clark, June 6, 2008 |
| 8. | Excerpt from Oracle Database Application Guide – Fundamentals, 10g Release 2 (10.2), B14251-01 |
| 9. | Excerpt from Oracle Application Server Containers for J2EE User's Guide 10g (9.0.4) B10322-01, ORCL00338416-00338661 |
| 10. | Parties' Proposed Element-by-Element Claim Construction Chart, June 30, 2008 |
| 11. | Excerpt from Oracle Application Server Web Cache Administrator's Guide, 10g Release 2 (10.1.2), B14046-05, ORCL00513887-00514436 |
| 12. | Excerpt from Oracle XML DB Developer's Guide 10g Release 2 (10.2), B14259-02. |
| 13. | Excerpt from Oracle Real Application Clusters Administration and Deployment Guide, 11g Release 1 (11.1), B28254-04, ORCL01972334-01972630 |
| 14. | What's New for Java DB, JDBC, and Database Web Servers in Oracle Database 10g, an Oracle White Paper, May 2005 |
| 15. | Excerpt from Oracle Database New Features Guide, 11g Release 1 (11.1) B28279-02 |
| 16. | Functional Specification for Runtime Load Balancing, RDBMS, 10gR2, ORCL01060148-01060193 |
| 17. | Excerpt from Plaintiff's Responses to Defendants' First Set of Interrogatories (1-6) |
| 18. | Oracle World, ORCL01050220-01050280 |

| Exh. | Document |
|---|---|
| 19. | Excerpt from Oracle Application Server 10g Concepts, 10g (9.0.4), B10375-02, ORCL00370687-00370914 |
| 20. | Oracle9i Application Server, An Oracle White Paper, September 2002, ORCL00010730-00010792 |
| 21. | Excerpt from Oracle Call Interface Programmer's Guide, 11g Release 1 (11.1), B28395-03 |
| 22. | Design Specification for OCI Runtime Connection Load Balancing in 11g, 12/14/08, ORCL01310468-01310498 |
| 23. | Functional/Design Specification for Runtime Work Request Load Balancing using RAC Service Metrics, JDBC, 10gR2, 5/1/04, ORCL01054844-01054858 |
| 24. | Oracle HTTP Server, mod_plsql User's Guide, 10g Release 2 (10.2), B14337-01 |
| 25. | Excerpt from deposition transcript of Paul C, Clark, July 11, 2008 |
| 26. | Excerpt from deposition transcript of Steve Harris, April 18, 2008 |
| 27. | Excerpt from deposition transcript of Pushkar Kapasi, February 27, 2008 |
| 28. | Excerpt from deposition transcript of Matthew Mayerson, April 15, 2008 |
| 29. | Excerpt from deposition transcript of Katrina Montinola, February 12, 2008 |
| 30. | Excerpt from deposition transcript of Mark Nelson, February 22, 2008 |
| 31. | Excerpt from deposition transcript of Douglas Surber, February 28, 2008 |
| 32. | Oracle10iAS Planning Update, ORCL01066383-01066413 |

**IN THE UNITED STATES DISTRICT COURT**
**FOR THE DISTRICT OF DELAWARE**

**CERTIFICATE OF SERVICE**

I, David E. Moore, hereby certify that on August 7, 2008, the attached document was electronically filed with the Clerk of the Court using CM/ECF which will send notification to the registered attorney(s) of record that the document has been filed and is available for viewing and downloading.

I further certify that on August 7, 2008, the attached document was Electronically Mailed to the following person(s):

Mary B. Graham
Morris, Nichols, Arsht & Tunnell LLP
1201 North Market Street
P.O. Box 1347
Wilmington, DE 19899
MGraham@MNAT.com

James G. Gilliland
Igor Shoiket
Townsend and Townsend and Crew LLP
Two Embarcadero Center
Eighth Floor
San Francisco, CA 94111-3834
OracleEpicrealm@townsend.com

Theodore T. Herhold
Robert J. Artuz
Eric M. Hutchins
Eric A. Mercer
Joseph A. Greco
Nitin Gupta
Townsend and Townsend and Crew LLP
379 Lytton Avenue
Palo Alto, CA 94301
OracleEpicrealm@townsend.com

Chad E. King
Townsend and Townsend and Crew LLP
1200 Seventeenth Street
Suite 2700
Denver, CO 80202
OracleEpicrealm@townsend.com

*/s/ David E. Moore*
Richard L. Horwitz
David E. Moore
Potter Anderson & Corroon LLP
Hercules Plaza – Sixth Floor
1313 North Market Street
P.O. Box 951
Wilmington, DE 19899-0951
(302) 984-6000
rhorwitz@potteranderson.com
dmoore@potteranderson.com

# Exhibit 1

US005894554A

# United States Patent [19]

Lowery et al.

[11] Patent Number: 5,894,554

[45] Date of Patent: Apr. 13, 1999

[54] **SYSTEM FOR MANAGING DYNAMIC WEB PAGE GENERATION REQUESTS BY INTERCEPTING REQUEST AT WEB SERVER AND ROUTING TO PAGE SERVER THEREBY RELEASING WEB SERVER TO PROCESS OTHER REQUESTS**

[75] Inventors: **Keith Lowery**, Richardson; **Andrew B. Levine**, Plano; **Ronald L. Howell**, Rowlett, all of Tex.

[73] Assignee: **InfoSpinner, Inc.**, Richardson, Tex.

[21] Appl. No.: **08/636,477**

[22] Filed: **Apr. 23, 1996**

[51] **Int. Cl.**[6] ..... **G06F 13/14**; G06F 13/20

[52] **U.S. Cl.** ..... **395/200.33**; 395/200.68; 395/200.75; 395/680; 707/10; 707/104

[58] **Field of Search** ..... 358/400; 395/800, 395/700, 200.68, 200.75, 200.53, 680, 200.33; 707/104, 10

[56] **References Cited**

U.S. PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| 4,866,706 | 9/1989 | Christophersen et al. | 370/85.7 |
| 5,341,499 | 8/1994 | Doragh | 395/700 |
| 5,392,400 | 2/1995 | Berkowitz et al. | 395/200 |
| 5,404,522 | 4/1995 | Carmon et al. | 395/650 |
| 5,404,523 | 4/1995 | DellaFera et al. | 395/650 |
| 5,404,527 | 4/1995 | Irwin et al. | 395/700 |
| 5,452,460 | 9/1995 | Distelberg et al. | 395/700 |
| 5,532,838 | 7/1996 | Barbari | 358/400 |
| 5,751,956 | 5/1998 | Kirsch | 395/200.33 |
| 5,761,673 | 6/1998 | Bookman et al. | 707/104 |

OTHER PUBLICATIONS

"Beyond the Web: Excavating the Real World Via Mosaic", Goldberg et al. Second International WWW, Oct. 17, 1994.

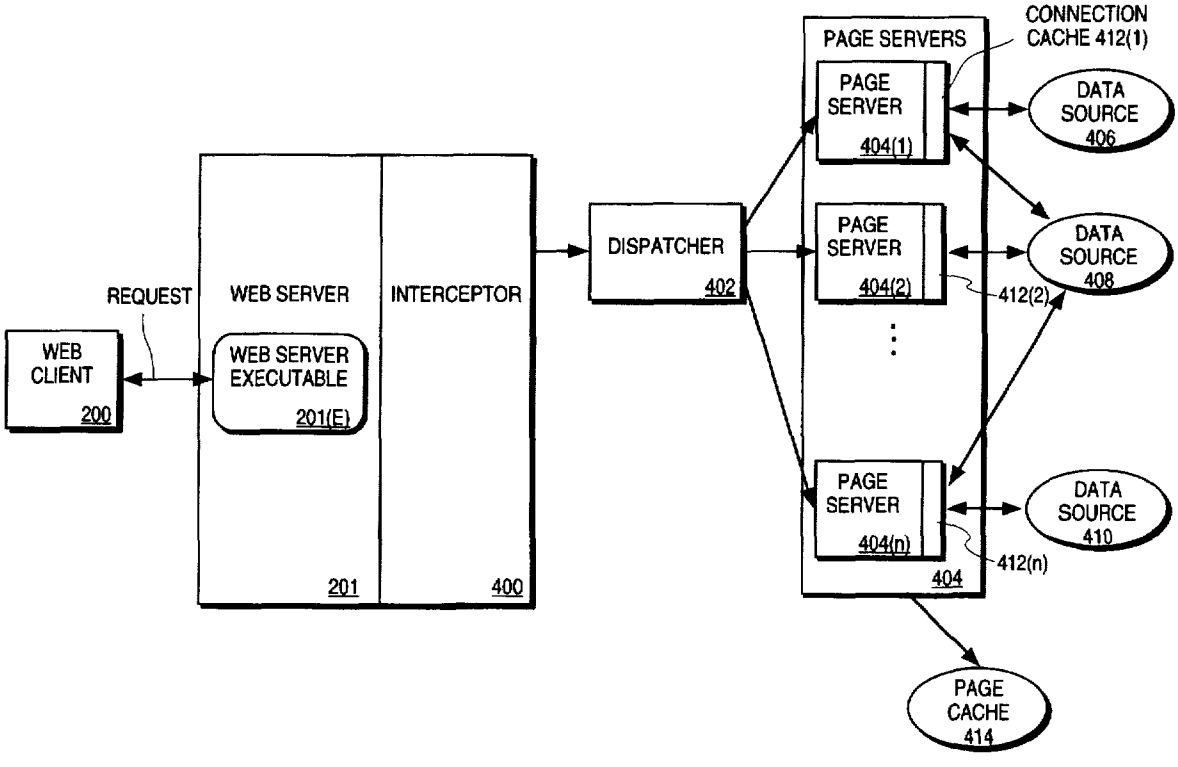
PCT International Search Report, Aug. 21, 1997.

*Primary Examiner*—Thomas C. Lee
*Assistant Examiner*—Rehana Perveen
*Attorney, Agent, or Firm*—Blakely, Sokoloff, Taylor & Zafman LLP

[57] **ABSTRACT**

The present invention teaches a method and apparatus for creating and managing custom Web sites. Specifically, one embodiment of the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests, processing the request, the processing being performed by the page server concurrently with the Web server, as the Web server processes the other requests, and dynamically generating a Web page in response to the request, the Web page including data dynamically retrieved from one or more data sources.

**11 Claims, 5 Drawing Sheets**



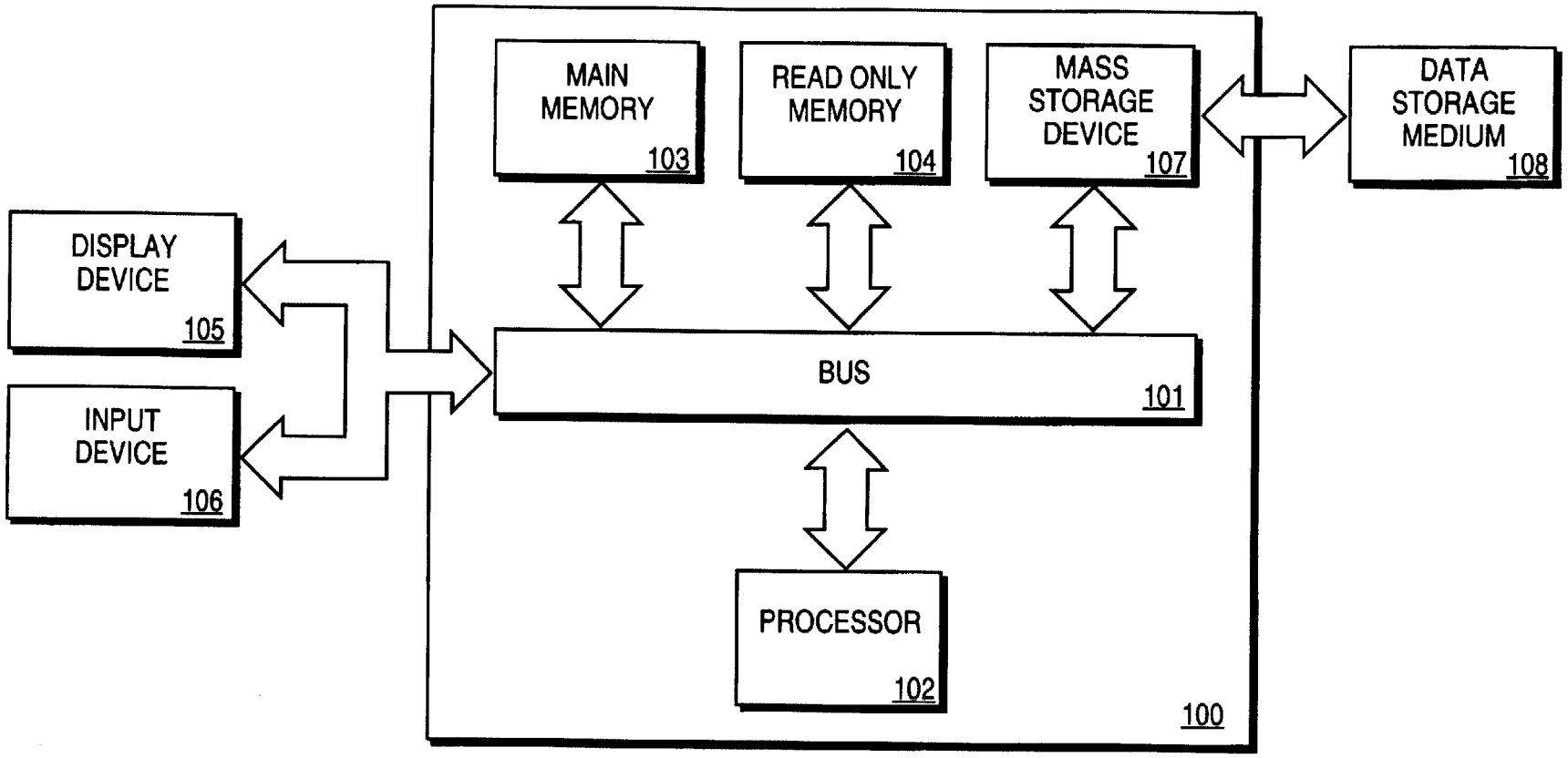

MAIN MEMORY
103
READ ONLY MEMORY
104
MASS STORAGE DEVICE
107
DATA STORAGE MEDIUM
108
DISPLAY DEVICE
105
INPUT DEVICE
106
BUS
101
PROCESSOR
102
100

FIG. 1

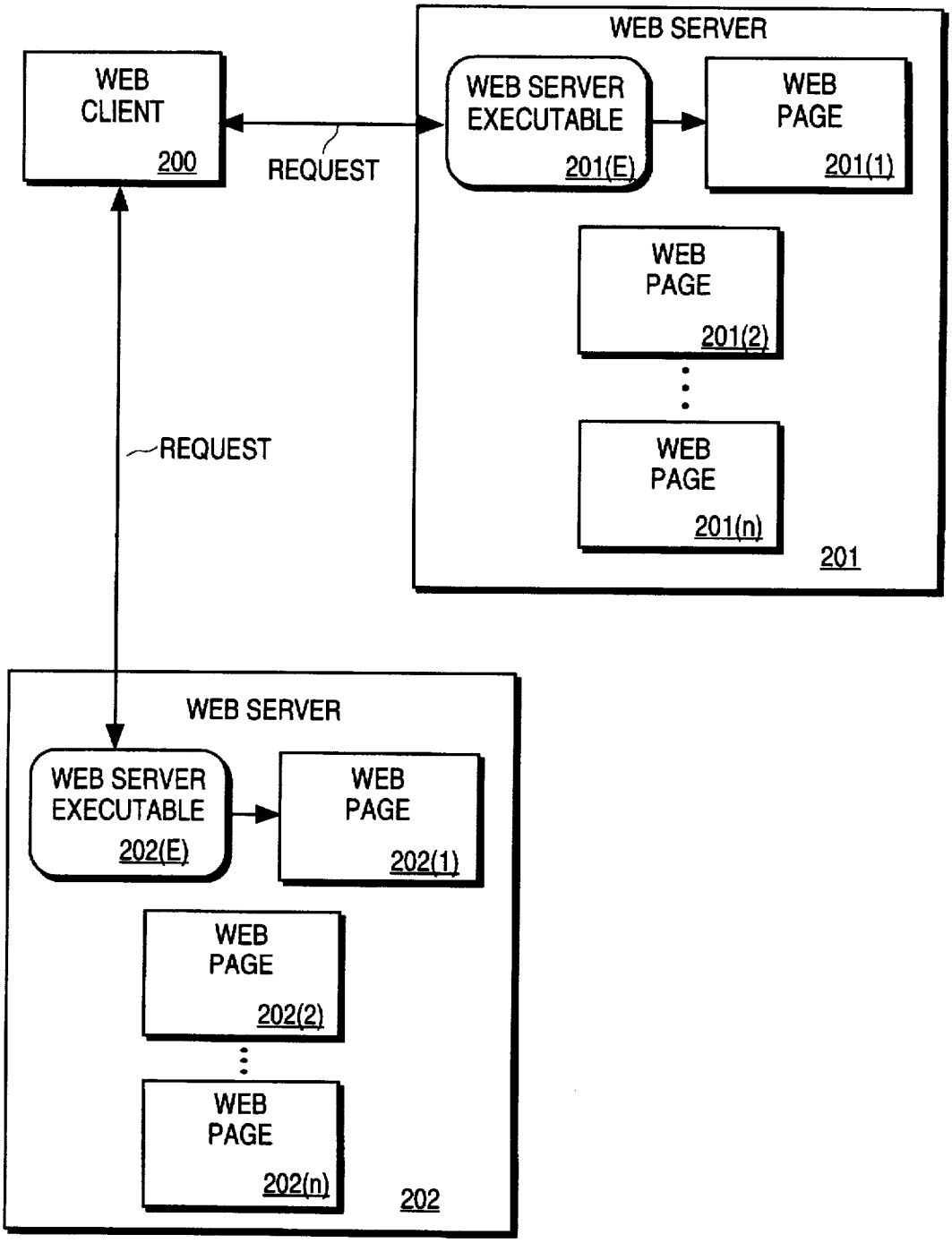



FIG. 2 (PRIOR ART)

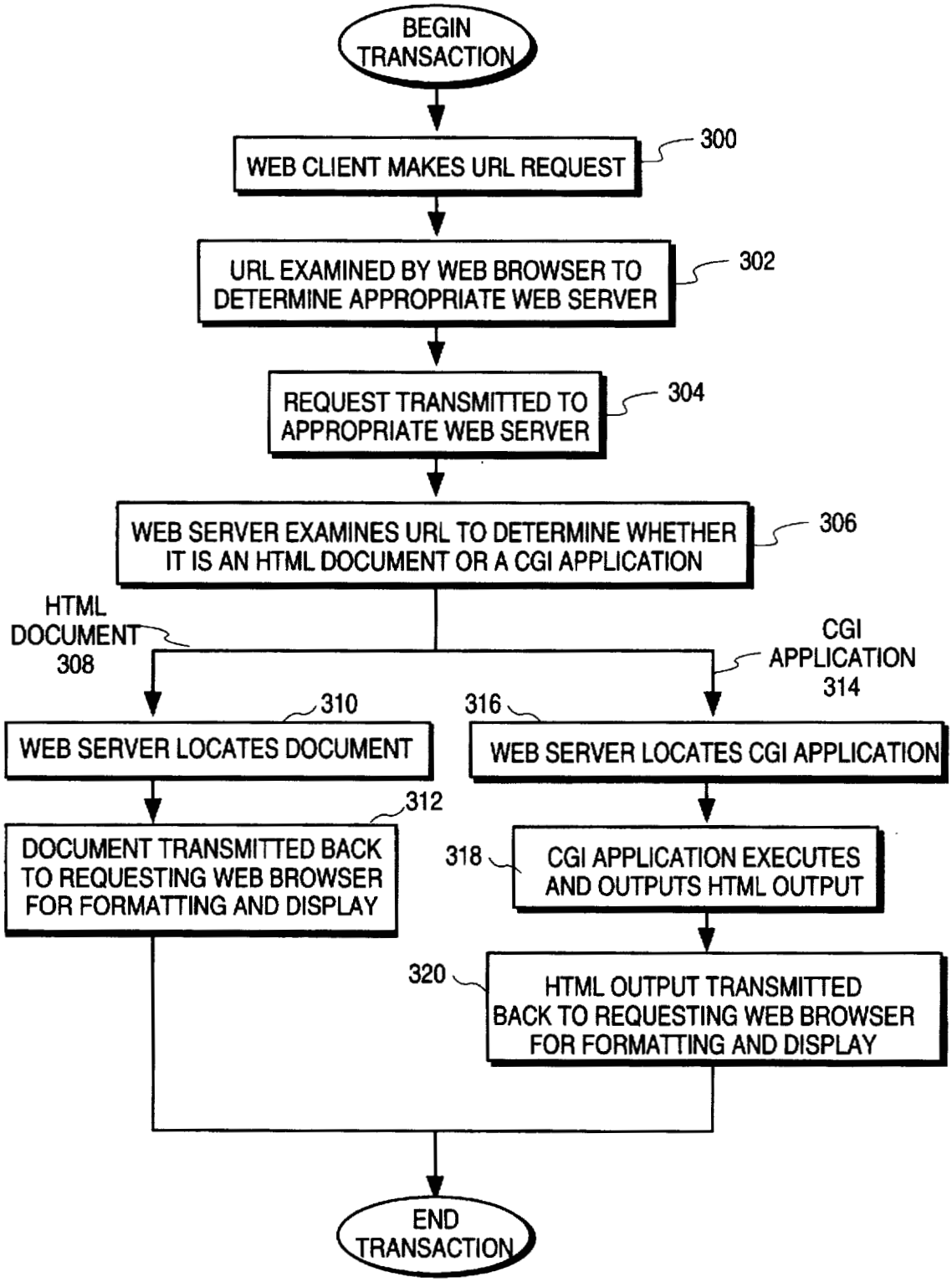



**FIG. 3** (PRIOR ART)

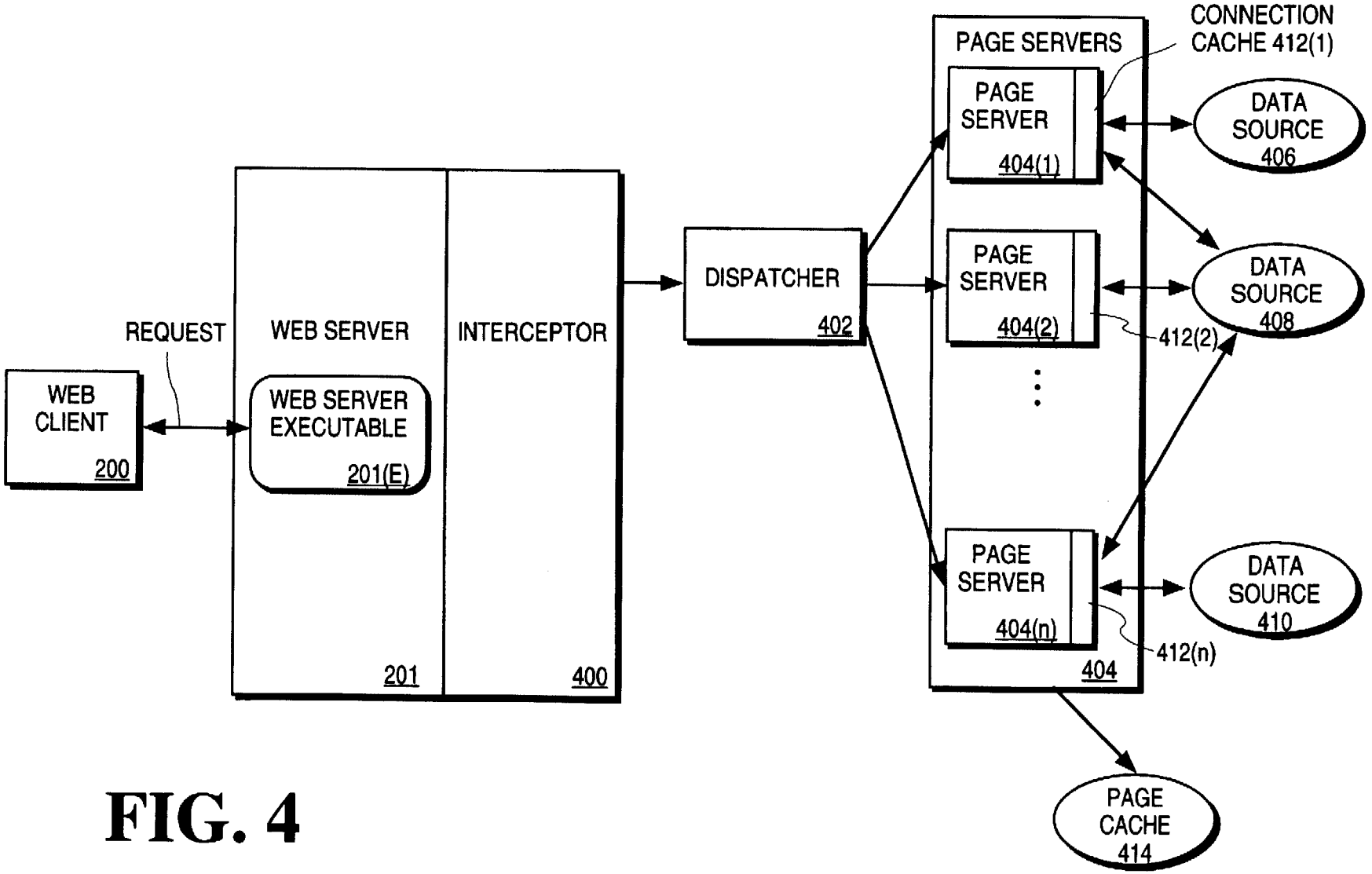

WEB CLIENT
200
REQUEST
WEB SERVER
WEB SERVER EXECUTABLE
201(E)
201
INTERCEPTOR
400
DISPATCHER
402
PAGE SERVERS
CONNECTION CACHE 412(1)
PAGE SERVER
404(1)
DATA SOURCE
406
PAGE SERVER
404(2)
412(2)
DATA SOURCE
408
PAGE SERVER
404(n)
412(n)
404
DATA SOURCE
410
PAGE CACHE
414

FIG. 4

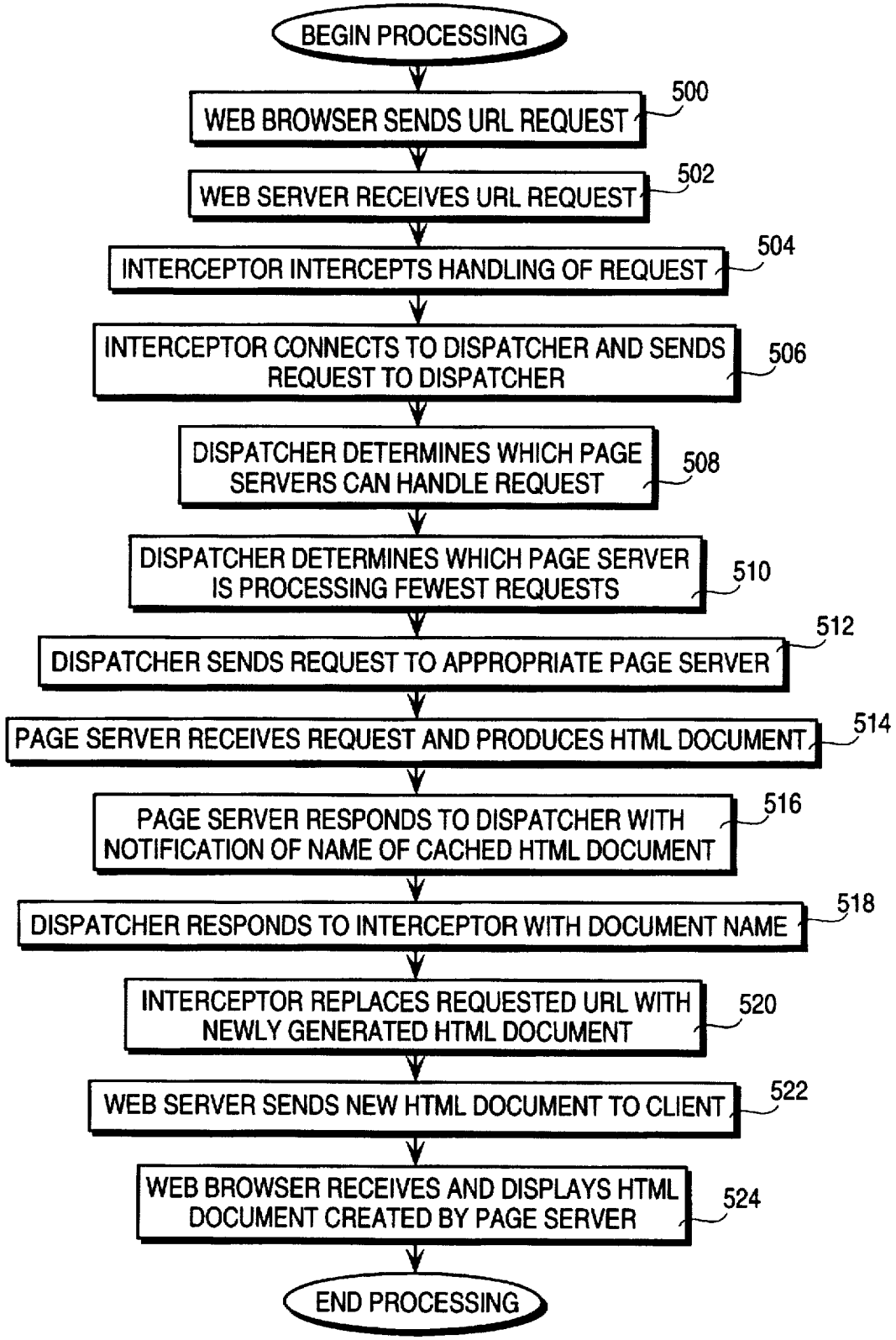

BEGIN PROCESSING
WEB BROWSER SENDS URL REQUEST
500
WEB SERVER RECEIVES URL REQUEST
502
INTERCEPTOR INTERCEPTS HANDLING OF REQUEST
504
INTERCEPTOR CONNECTS TO DISPATCHER AND SENDS REQUEST TO DISPATCHER
506
DISPATCHER DETERMINES WHICH PAGE SERVERS CAN HANDLE REQUEST
508
DISPATCHER DETERMINES WHICH PAGE SERVER IS PROCESSING FEWEST REQUESTS
510
DISPATCHER SENDS REQUEST TO APPROPRIATE PAGE SERVER
512
PAGE SERVER RECEIVES REQUEST AND PRODUCES HTML DOCUMENT
514
PAGE SERVER RESPONDS TO DISPATCHER WITH NOTIFICATION OF NAME OF CACHED HTML DOCUMENT
516
DISPATCHER RESPONDS TO INTERCEPTOR WITH DOCUMENT NAME
518
INTERCEPTOR REPLACES REQUESTED URL WITH NEWLY GENERATED HTML DOCUMENT
520
WEB SERVER SENDS NEW HTML DOCUMENT TO CLIENT
522
WEB BROWSER RECEIVES AND DISPLAYS HTML DOCUMENT CREATED BY PAGE SERVER
524
END PROCESSING

FIG. 5

# SYSTEM FOR MANAGING DYNAMIC WEB PAGE GENERATION REQUESTS BY INTERCEPTING REQUEST AT WEB SERVER AND ROUTING TO PAGE SERVER THEREBY RELEASING WEB SERVER TO PROCESS OTHER REQUESTS

## FIELD OF THE INVENTION

The present invention relates to the field of Internet technology. Specifically, the present invention relates to the creation and management of custom World Wide Web sites.

## DESCRIPTION OF RELATED ART

The World Wide Web (the Web) represents all of the computers on the Internet that offer users access to information on the Internet via interactive documents or Web pages. These Web pages contain hypertext links that are used to connect any combination of graphics, audio, video and text, in a non-linear, non-sequential manner. Hypertext links are created using a special software language known as HyperText Mark-Up Language (HTML).

Once created, Web pages reside on the Web, on Web servers or Web sites. A Web site can contain numerous Web pages. Web client machines running Web browsers can access these Web pages at Web sites via a communications protocol known as HyperText Transport Protocol (HTTP). Web browsers are software interfaces that run on World Wide Web clients to allow access to Web sites via a simple user interface. A Web browser allows a Web client to request a particular Web page from a Web site by specifying a Uniform Resource Locator (URL). A URL is a Web address that identifies the Web page and its location on the Web. When the appropriate Web site receives the URL, the Web page corresponding to the requested URL is located, and if required, HTML output is generated. The HTML output is then sent via HTTP to the client for formatting on the client's screen.

Although Web pages and Web sites are extremely simple to create, the proliferation of Web sites on the Internet highlighted a number of problems. The scope and ability of a Web page designer to change the content of the Web page was limited by the static nature of Web pages. Once created, a Web page remained static until it was manually modified. This in turn limited the ability of Web site managers to effectively manage their Web sites.

The Common Gateway Interface (CGI) standard was developed to resolve the problem of allowing dynamic content to be included in Web pages. CGI "calls" or procedures enable applications to generate dynamically created HTML output, thus creating Web pages with dynamic content. Once created, these CGI applications do not have to be modified in order to retrieve "new" or dynamic data. Instead, when the Web page is invoked, CGI "calls" or procedures are used to dynamically retrieve the necessary data and to generate a Web page.

CGI applications also enhanced the ability of Web site administrators to manage Web sites. Administrators no longer have to constantly update static Web pages. A number of vendors have developed tools for CGI based development, to address the issue of dynamic Web page generation. Companies like Spider™ and Bluestone™, for example, have each created development tools for CGI-based Web page development. Another company, Haht Software™, has developed a Web page generation tool that uses a BASIC-like scripting language, instead of a CGI scripting language.



Tools that generate CGI applications do not, however, resolve the problem of managing numerous Web pages and requests at a Web site. For example, a single company may maintain hundreds of Web pages at their Web site. Current Web server architecture also does not allow the Web server to efficiently manage the Web page and process Web client requests. Managing these hundreds of Web pages in a coherent manner and processing all requests for access to the Web pages is thus a difficult task. Existing development tools are limited in their capabilities to facilitate dynamic Web page generation, and do not address the issue of managing Web requests or Web sites.

## SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a method and apparatus for creating and managing custom Web sites. Specifically, the present invention claims a method and apparatus for managing dynamic web page generation requests.

In one embodiment, the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests, processing the request, the processing being performed by the page server concurrently with the Web server, as the Web server processes the other requests, and dynamically generating a Web page in response to the request, the Web page including data dynamically retrieved from one or more data sources. Other embodiments also include connection caches to the one or more data sources, page caches for each page server, and custom HTML extension templates for configuring the Web page.

Other objects, features and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a typical computer system in which the present invention operates.

FIG. **2** illustrates a typical prior art Web server environment.

FIG. **3** illustrates a typical prior art Web server environment in the form of a flow diagram.

FIG. **4** illustrates one embodiment of the presently claimed invention.

FIG. **5** illustrates the processing of a Web browser request in the farm of a flow diagram, according to one embodiment of the presently claimed invention.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention relates to a method and apparatus for creating and managing custom Web sites. In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent to one of ordinary skill in the art, however, that these specific details need not be used to practice the present invention. In other instances, well-known structures, interfaces and processes have not been shown in detail in order not to unnecessarily obscure the present invention.

FIG. **1** illustrates a typical computer system **100** in which the present invention operates. The preferred embodiment of

the present invention is implemented on an IBM™ Personal Computer manufactured by IBM Corporation of Armonk, N.Y. An alternate embodiment may be implemented on an RS/6000™ Workstation manufactured by IBM Corporation of Armonk, N.Y. It will be apparent to those of ordinary skill in the art that other computer system architectures may also be employed.

In general, such computer systems as illustrated by FIG. 1 comprise a bus **101** for communicating information, a processor **102** coupled with the bus **101** for processing information, main memory **103** coupled with the bus **101** for storing information and instructions for the processor **102**, a read-only memory **104** coupled with the bus **101** for storing static information and instructions for the processor **102**, a display device **105** coupled with the bus **101** for displaying information for a computer user, an input device **106** coupled with the bus **101** for communicating information and command selections to the processor **102**, and a mass storage device **107**, such as a magnetic disk and associated disk drive, coupled with the bus **101** for storing information and instructions. A data storage medium **108** containing digital information is configured to operate with mass storage device **107** to allow processor **102** access to the digital information on data storage medium **108** via bus **101**.

Processor **102** may be any of a wide variety of general purpose processors or microprocessors such as the Pentium™ microprocessor manufactured by Intel™ Corporation or the RS/6000™ processor manufactured by IBM Corporation. It will be apparent to those of ordinary skill in the art, however, that other varieties of processors may also be used in a particular computer system. Display device **105** may be a liquid crystal device, cathode ray tube (CRT), or other suitable display device. Mass storage device **107** may be a conventional hard disk drive, floppy disk drive, CD-ROM drive, or other magnetic or optical data storage device for reading and writing information stored on a hard disk, a floppy disk, a CD-ROM a magnetic tape, or other magnetic or optical data storage medium. Data storage medium **108** may be a hard disk, a floppy disk, a CD-ROM, a magnetic tape, or other magnetic or optical data storage medium.

In general, processor **102** retrieves processing instructions and data from a data storage medium **108** using mass storage device **107** and downloads this information into random access memory **103** for execution. Processor **102**, then executes an instruction stream from random access memory **103** or read-only memory **104**. Command selections and information input at input device **106** are used to direct the flow of instructions executed by processor **102**. Equivalent input device **106** may also be a pointing device such as a conventional mouse or trackball device. The results of this processing execution are then displayed on display device **105**.

The preferred embodiment of the present invention is implemented as a software module, which may be executed on a computer system such as computer system **100** in a conventional manner. Using well known techniques, the application software of the preferred embodiment is stored on data storage medium **108** and subsequently loaded into and executed within computer system **100**. Once initiated, the software of the preferred embodiment operates in the manner described below.

FIG. **2** illustrates a typical prior art Web server environment. Web client **200** can make URL requests to Web server **201** or Web server **202**. Web servers **201** and **202** include Web server executables, **201**(E) and **202**(E) respectively, that perform the processing of Web client requests. Each Web server may have a number of Web pages **201**(1)–(n) and **202**(1)–(n). Depending on the URL specified by the Web client **200**, the request may be routed by either Web server executable **201**(E) to Web page **201** (1), for example, or from Web server executable **202**(E) to Web page **202** (1). Web client **200** can continue making URL requests to retrieve other Web pages. Web client **200** can also use hyperlinks within each Web page to "jump" to other Web pages or to other locations within the same Web page.

FIG. **3** illustrates this prior art Web server environment in the form of a flow diagram. In processing block **300**, the Web client makes a URL request. This URL request is examined by the Web browser to determine the appropriate Web server to route the request to in processing block **302**. In processing block **304** the request is then transmitted from the Web browser to the appropriate Web server, and in processing block **306** the Web server executable examines the URL to determine whether it is a HTML document or a CGI application. If the request is for an HTML document **308**, then the Web server executable locates the document in processing block **310**. The document is then transmitted back through the requesting Web browser for formatting and display in processing block **312**.

If the URL request is for a CGI application **314**, however, the Web server executable locates the CGI application in processing block **316**. The CGI application then executes and outputs HTML output in processing block **318** and finally, the HTML output is transmitted back to requesting Web browser for formatting and display in processing block **320**.

This prior art Web server environment does not, however, provide any mechanism for managing the Web requests or the Web sites. As Web sites grow, and as the number of Web clients and requests increase, Web site management becomes a crucial need.

For example, a large Web site may receive thousands of requests or "hits" in a single day. Current Web servers process each of these requests on a single machine, namely the Web server machine. Although these 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. As such, the Web executable thread may hand off a request to a processing thread, but both threads will still have to be handled by the processor on the Web server machine. When numerous requests are being simultaneously processed by multiple threads on a single machine, the Web server can slow down significantly and become highly inefficient. The claimed invention addresses this need by utilizing a partitioned architecture to facilitate the creation and management of custom Web sites and servers.

FIG. **4** illustrates one embodiment of the presently claimed invention. Web client **200** issues a URL request that is processed to determined proper routing. In this embodiment, the request is routed to Web server **201**. Instead of Web server executable **201**(E) processing the URL request, however, Interceptor **400** intercepts the request and routes it to Dispatcher **402**. In one embodiment, Interceptor **400** resides on the Web server machine as an extension to Web server **201**. This embodiment is appropriate for Web servers such as Netsite™ from Netscape, that support such extensions. A number of public domain Web servers, such as NCSA™ from the National Center for Supercomputing Applications at the University of Illinois, Urbana-Champaign, however, do not provide support for

this type of extension. Thus, in an alternate embodiment, Interceptor **400** is an independent module, connected via an "intermediate program" to Web server **201**. This intermediate program can be a simple CGI application program that connects Interceptor **400** to Web server **201**. Alternate intermediate programs the perform the same functionality can also be implemented.

In one embodiment of the invention, Dispatcher **402** resides on a different machine than Web server **201**. This embodiment overcomes the limitation described above, in prior art Web servers, wherein all processing is performed by the processor on a single machine. By routing the request to Dispatcher **402** residing on a different machine than the Web server executable **201**(E), the request can then be processed by a different processor than the Web server executable **201**(E). Web server executable **201**(E) is thus free to continue servicing client requests on Web server **201** while the request is processed "off-line," at the machine on which Dispatcher **402** resides.

Dispatcher **402** can, however, also reside on the same machine as the Web server. The Web site administrator has the option of configuring Dispatcher **402** on the same machine as Web server **201**, taking into account a variety of factors pertinent to a particular Web site, such as the size of the Web site, the number of Web pages and the number of hits at the Web site. Although this embodiment will not enjoy the advantage described above, namely off-loading the processing of Web requests from the Web server machine, the embodiment does allow flexibility for a small Web site to grow. For example, a small Web site administrator can use a single machine for both Dispatcher **402** and Web server **201** initially, then off-load Dispatcher **402** onto a separate machine as the Web site grows. The Web site can thus take advantage of other features of the present invention regardless of whether the site has separate machines configured as Web servers and dispatchers.

Dispatcher **402** receives the intercepted request and then dispatches the request to one of a number of Page servers **404** (1)–(n). For example, if Page server **404** (1) receives the dispatched request, it processes the request and retrieves the data from an appropriate data source, such as data source **406**, data source **408**, or data source **410**. Data sources, as used in the present application, include databases, spreadsheets, files and any other type of data repository. Page server **404** (1) can retrieve data from more than one data source and incorporate the data from these multiple data sources in a single Web page.

In one embodiment, each Page server **404**(1)–(n) resides on a separate machine on the network to distribute the processing of the request. Dispatcher **402** maintains a variety of information regarding each Page server on the network, and dispatches requests based on this information. For example, Dispatcher **402** retains dynamic information regarding the data sources that any given Page server can access. Dispatcher **402** thus examines a particular request and determines which Page servers can service the URL request. Dispatcher **402** then hands off the request to the appropriate Page server.

For example, if the URL request requires financial data from data source **408**, dispatcher **402** will first examine an information list. Dispatcher **402** may determine that Page server **404**(3), for example, has access to the requisite data in data source **408**. Dispatcher **402** will thus route the URL request to Page server **404**(3). This "connection caching" functionality is described in more detail below, under the heading "Performance."

Alternately, Dispatcher **402** also has the ability to determine whether a particular Page server already has the necessary data cached in the Page server's page cache (described in more detail below, under the heading "Performance"). Dispatcher **402** may thus determine that Page server **404**(1) and **404**(2) are both logged into Data source **408**, but that Page server **404**(2) has the financial information already cached in Page server **404**(2)'s page cache. In this case, Dispatcher **402** will route the URL request to Page server **404**(2) to more efficiently process the request.

Finally, Dispatcher **402** may determine that a number or all Page servers **404**(1)–(n) are logged into Data source **408**. In this scenario, Dispatcher **402** can examine the number of requests that each Page server is servicing and route the request to the least busy page server. This "load balancing" capability can significantly increase performance at a busy Web site and is discussed in more detail below, under the heading "Scalability".

If, for example, Page server **404**(2), receives the request, Page server **404**(2) will process the request. While Page server **404**(2) is processing the request, Web server executable **201**(E) can concurrently process other Web client requests. This partitioned architecture thus allows both Page server **404**(2) and Web server executable **201**(E) to simultaneously process different requests, thus increasing the efficiency of the Web site. Page server **404**(2) dynamically generates a Web page in response to the Web client request, and the dynamic Web page is then either transmitted back to requesting Web client **200** or stored on a machine that is accessible to Web server **201**, for later retrieval.

One embodiment of the claimed invention also provides a Web page designer with HTML extensions, or "dyna" tags. These dyna tags provide customized HTML functionality to a Web page designer, to allow the designer to build customized HTML templates that specify the source and placement of retrieved data. For example, in one embodiment, a "dynatext" HTML extension tag specifies a data source and a column name to allow the HTML template to identify the data source to log into and the column name from which to retrieve data. Alternatively, "dyna-anchor" tags allow the designer to build hyperlink queries while "dynablock" tags provide the designer with the ability to iterate through blocks of data. Page servers use these HTML templates to create dynamic Web pages. Then, as described above, these dynamic Web pages are either transmitted back to requesting Web client **200** or stored on a machine that is accessible to Web server **201**, for later retrieval.

The presently claimed invention provides numerous advantages over prior art Web servers, including advantages in the areas of performance, security, extensibility and scalability.

## Performance

One embodiment of the claimed invention utilizes connection caching and page caching to improve performance. Each Page server can be configured to maintain a cache of connections to numerous data sources. For example, as illustrated in FIG. **4**, Page server **404**(1) can retrieve data from data source **406**, data source **408** or data source **410**. Page server **404**(1) can maintain connection cache **412**(1), containing connections to each of data source **406**, data source **408** and data source **410**, thus eliminating connect times from the Page servers to those data sources.

Additionally, another embodiment of the present invention supports the caching of finished Web pages, to optimize

the performance of the data source being utilized. This "page caching" feature, illustrated in FIG. **4** as Page cache **414**, allows the Web site administrator to optimize the performance of data sources by caching Web pages that are repeatedly accessed. Once the Web page is cached, subsequent requests or "hits" will utilize the cached Web page rather than re-accessing the data source. This can radically improve the performance of the data source.

### Security

The present invention allows the Web site administrator to utilize multiple levels of security to manage the Web site. In one embodiment, the Page server can utilize all standard encryption and site security features provided by the Web server. In another embodiment, the Page server can be configured to bypass connection caches **412**(1)–(n), described above, for a particular data source and to require entry of a user-supplied identification and password for the particular data source the user is trying to access.

Additionally, another embodiment of the presently claimed invention requires no real-time access of data sources. The Web page caching ability, described above, enables additional security for those sites that want to publish non-interactive content from internal information systems, but do not want real-time Internet accessibility to those internal information systems. In this instance, the Page server can act as a "replication and staging gent" and create Web pages in batches, rather than in real-time. These "replicated" Web pages are then "staged" for access at a later time, and access o the Web pages in this scenario is possible even if the Page server and dispatcher are not present later.

In yet another embodiment, the Page server can make a single pass through a Web library, and compile a Web site that exists in the traditional form of separately available files. A Web library is a collection of related Web books and Web pages. More specifically, the Web library is a hierarchical organization of Web document templates, together with all the associated data source information. Information about an entire Web site is thus contained in a single physical file, thus simplifying the problem of deploying Web sites across multiple Page servers. The process of deploying the Web site in this embodiment is essentially a simple copy of a single file.

### Extensibility

One embodiment of the present invention provides the Web site administrator with Object Linking and Embedding (OLE) 2.0 extensions to extend the page creation process. These OLE 2.0 extensions also allow information submitted over the Web to be processed with user-supplied functionality. Utilizing development tools such as Visual Basic, Visual C++ or PowerBuilder that support the creation of OLE 2.0 automation, the Web site administrator can add features and modify the behavior of the Page servers described above. This extensibility allows one embodiment of the claimed invention to be incorporated with existing technology to develop an infinite number of custom web servers.

For example, OLE 2.0 extensions allow a Web site administrator to encapsulate existing business rules in an OLE 2.0 automation interface, to be accessed over the Web. One example of a business rule is the steps involved in the payoff on an installment or mortgage loan. The payoff may involve, for example, taking into account the current balance, the date and the interest accrued since the last payment. Most organizations already have this type of business rule implemented using various applications, such as Visual Basic for client-server environments, or CICS programs on mainframes. If these applications are OLE 2.0 compliant, the Page server "dynaobject" HTML extension tag can be used to encapsulated the application in an OLE 2.0 automation interface. The Page server is thus extensible, and can incorporate the existing application with the new Page server functionality.

### Scalability

One embodiment of the claimed invention allows "plug and play" scalability. As described above, referring to FIG. **4**, Dispatcher **402** maintains information about all the Page servers configured to be serviced by Dispatcher **402**. Any number of Page servers can thus be "plugged" into the configuration illustrated in FIG. **4**, and the Page servers will be instantly activated as the information is dynamically updated in Dispatcher **402**. The Web site administrator can thus manage the overhead of each Page server and modify each Page server's load, as necessary, to improve performance. In this manner, each Page server will cooperate with other Page servers within a multi-server environment. Dispatcher **402** can examine the load on each Page server and route new requests according to each Page server's available resources. This "load-balancing" across multiple Page servers can significantly increase a Web site's performance.

FIG. **5** illustrates the processing of a Web browser request in the form of a flow diagram, according to one embodiment of the presently claimed invention. A Web browser sends a URL request to a Web server in processing block **500**. In processing block **502**, the Web server receives the URL request, and an interceptor then intercepts the handling of the request in processing block **504**. The interceptor connects to a dispatcher and sends the URL request to the dispatcher in processing block **506**. In processing block **508**, the dispatcher determines which Page servers can handle the request. The dispatcher also determines which Page server is processing the fewest requests in processing block **510**, and in processing block **512**, the dispatcher sends the URL request to an appropriate Page server. The Page server receives the request and produces an HTML document in processing block **514**. The Page server then responds to the dispatcher with notification of the name of the cached HTML document in processing block **516**. In processing block **518**, the dispatcher responds to the interceptor with the document name, and the interceptor then replaces the requested URL with the newly generated HTML document in processing block **520**. The Web server then sends the new HTML document to the requesting client in processing block **522**. Finally, the Web browser receives and displays the HTML document created by the Page server at processing block **524**.

Thus, a method and apparatus for creating and managing custom Web sites is disclosed. These specific arrangements and methods described herein are merely illustrative of the principles of the present invention. Numerous modifications in form and detail may be made by those of ordinary skill in the art without departing from the scope of the present invention. Although this invention has been shown in relation to a particular preferred embodiment, it should not be considered so limited. Rather, the present invention is limited only by the scope of the appended claims.

We claim:

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

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

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

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

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

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

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

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

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

**6**. The computer-implemented method in claim **3** wherein said step of dynamically generating said Web page includes the step of maintaining a page cache containing said Web page.

**7**. The computer-implemented method in claim **3** wherein said page server includes custom HTML extension templates for configuring said Web page.

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

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

one or more data sources;

a page server having a processing means;

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

a second computer system including means for receiving said request from said first computer, said second computer system also including a router, said router routing said request from said second computer system to said page server, wherein said routing further includes intercepting said request at said second computer, routing said request from said second computer to a dispatcher, and dispatching said request to said page server said page server receiving said request and releasing said second computer system to process other requests, said page server processing means processing said request and dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from said one or more data sources.

**10**. The networked system in claim **9** wherein said router in said second computer system includes:

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

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

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

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

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

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

* * * * *

# Exhibit 2

US006415335B1

(12) **United States Patent**
**Lowery et al.**

(10) **Patent No.: US 6,415,335 B1**
(45) **Date of Patent: *Jul. 2, 2002**

(54) **SYSTEM AND METHOD FOR MANAGING DYNAMIC WEB PAGE GENERATION REQUESTS**

(75) Inventors: **Keith Lowery**, Richardson; **Andrew B. Levine**, Plano; **Ronald L. Howell**, Rowlett, all of TX (US)

(73) Assignee: **epicRealm Operating Inc.**, Richardson, TX (US)

( * ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **09/234,048**

(22) Filed: **Jan. 19, 1999**

**Related U.S. Application Data**

(62) Division of application No. 08/636,477, filed on Apr. 23, 1996, now Pat. No. 5,894,554.

(51) **Int. Cl.**[7] ..... **G06F 13/14**; G06F 13/20
(52) **U.S. Cl.** ..... **710/5**; 710/7; 709/219; 709/223; 709/238
(58) **Field of Search** ..... 709/238, 223, 709/219; 710/5, 7, 20–21

(56) **References Cited**

U.S. PATENT DOCUMENTS

| | | | | |
|---|---|---|---|---|
| 4,866,706 | A | | 9/1989 | Christophersen et al. .. 370/85.7 |
| 5,341,499 | A | | 8/1994 | Doragh ..... 395/700 |
| 5,392,400 | A | | 2/1995 | Berkowitz et al. ..... 395/200 |
| 5,404,522 | A | | 4/1995 | Carmon et al. ..... 395/650 |
| 5,404,523 | A | | 4/1995 | DellaFera et al. ..... 395/650 |
| 5,404,527 | A | * | 4/1995 | Irwin et al. ..... 395/700 |
| 5,452,460 | A | | 9/1995 | Distelberg et al. ..... 395/700 |
| 5,532,838 | A | | 7/1996 | Barbari ..... 358/400 |
| 5,701,463 | A | * | 12/1997 | Malcolm ..... 395/610 |
| 5,751,956 | A | | 5/1998 | Kirsch ..... 395/200.33 |
| 5,752,246 | A | * | 5/1998 | Rogers et al. ..... 707/10 |
| 5,754,772 | A | * | 5/1998 | Leaf ..... 395/200.33 |
| 5,761,673 | A | * | 6/1998 | Bookman et al. ..... 707/104 |
| 5,774,660 | A | | 6/1998 | Brendel et al. ..... 395/200.31 |
| 5,774,668 | A | | 6/1998 | Choquier et al. ..... 395/200.53 |

OTHER PUBLICATIONS

Hoffner, 'Inter–operability and distributed application platform design', Web URL:http://www.ansa.co.uk/, 1995, pp. 342–356.*

Mourad et al, 'Scalable Web Server Architectures', IEEE, Jun. 1997, pp. 12–16.*

Dias et al, 'A Scalable and Highly Available Web Server', IEEE, 1996, pp. 85–92.*

'Single System Image and Load Balancing for Network Access to a Loosely Coupled Complex', IBM TDB, vol. 34, Feb. 1992, pp. 464–467.*

Dias, Daniel M., et al.; A Scalable and Highly Available Web Server; IBM Research Division; T.J. Watson Research Center; 7 pages.

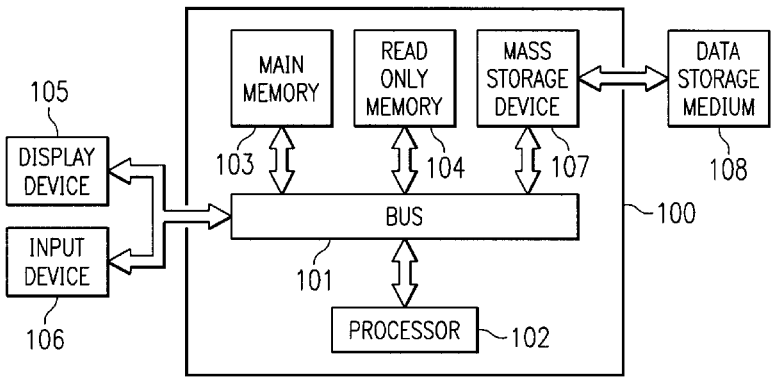
(List continued on next page.)

*Primary Examiner*—Jeffrey Gaffin
*Assistant Examiner*—Rehana Perveen
(74) *Attorney, Agent, or Firm*—Baker Botts L.L.P.

(57) **ABSTRACT**

The present invention teaches a method and apparatus for creating and managing custom Web sites. Specifically, one embodiment of the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests, processing the request, the processing being performed by the page server concurrently with the Web server, as the Web server processes the other requests, and dynamically generating a Web page in response to the request, the Web page including data dynamically retrieved from one or more data sources.

**29 Claims, 4 Drawing Sheets**



OTHER PUBLICATIONS

Andresen, Daniel, Et Al.; Scalability Issues for High Performance Digital Libraries on the World Wide Web; Department of Computer Science; University of California at Santa Barbara; 10 pages.

Andresen, Daniel, Et Al.; SWEB: Towards a Scalable World Wide Web Server on Multicomputers; Department of Computer Science; University of California at Santa Barbara; 7 pages.

Holmedahl, Vegard; Et Al.; Cooperative Caching of Dynamic Content on a Distributed Web Server; Department of Computer Science; University of California at Santa Barbara; 8 pages.

Overson, Nicole; NeXT Ships WebObjects—On Time—As Promised; Deja.com: NeXT Ships WebObjects—On Time—As Promishttp://X28..deja.com/=dnc/ST_m=ps...EXT=927585438. 1744765032&hitnum=33.

International Search Report; 7 pages; dated Aug. 21, 1997.

Birman, Kenneth P. and van Renesse, Robbert; Software for Reliable Networks; Scientific American; May 1996; pp. 64–69.

"Beyond the Web: Excavating the Real World Via Mosaic"; Goldberg et al.; Second International WWW Conference; Oct. 17, 1994.

* cited by examiner

FIG. 1

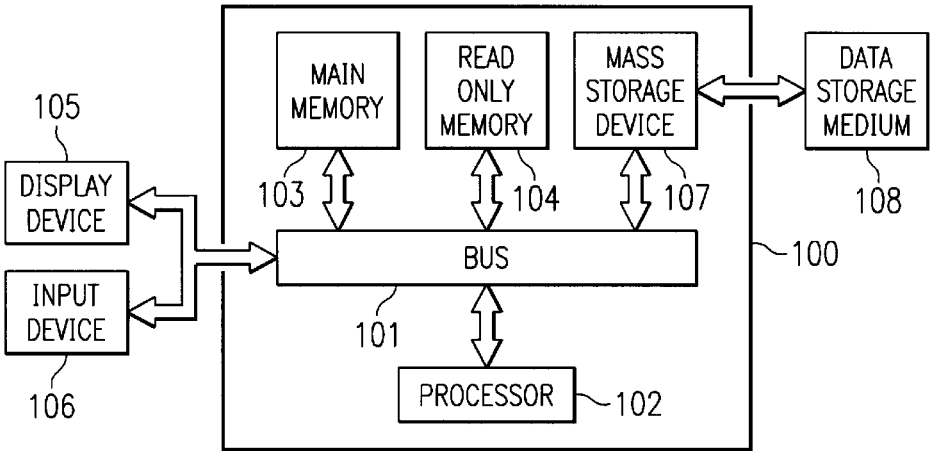



FIG. 2
(PRIOR ART)

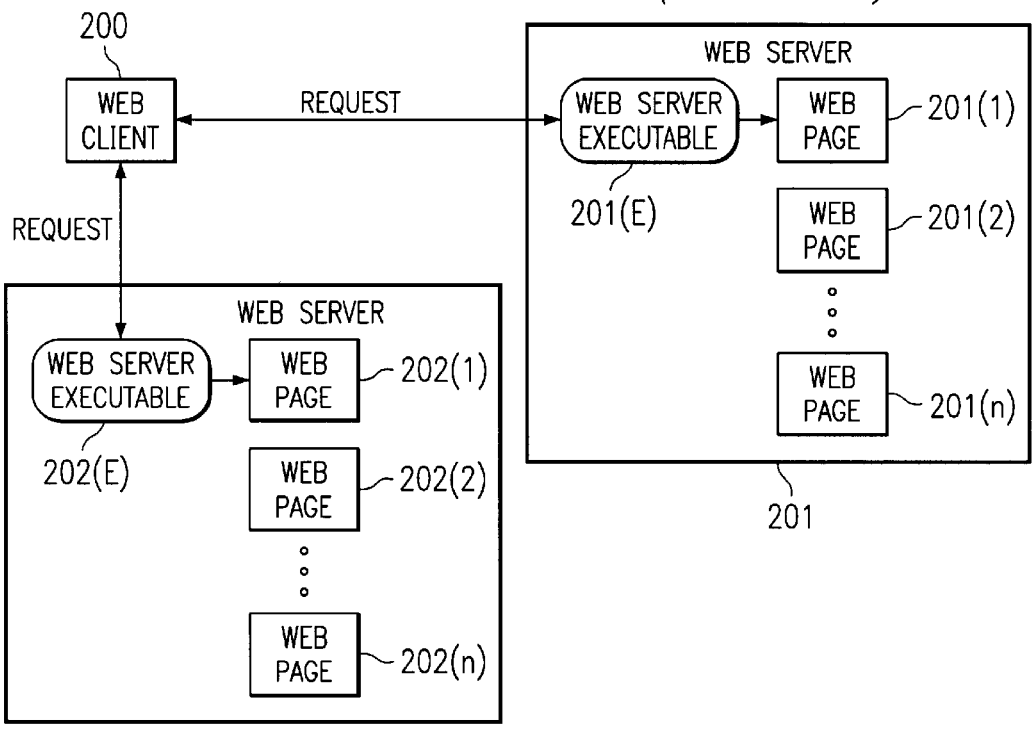


*FIG. 3*
*(PRIOR ART)*

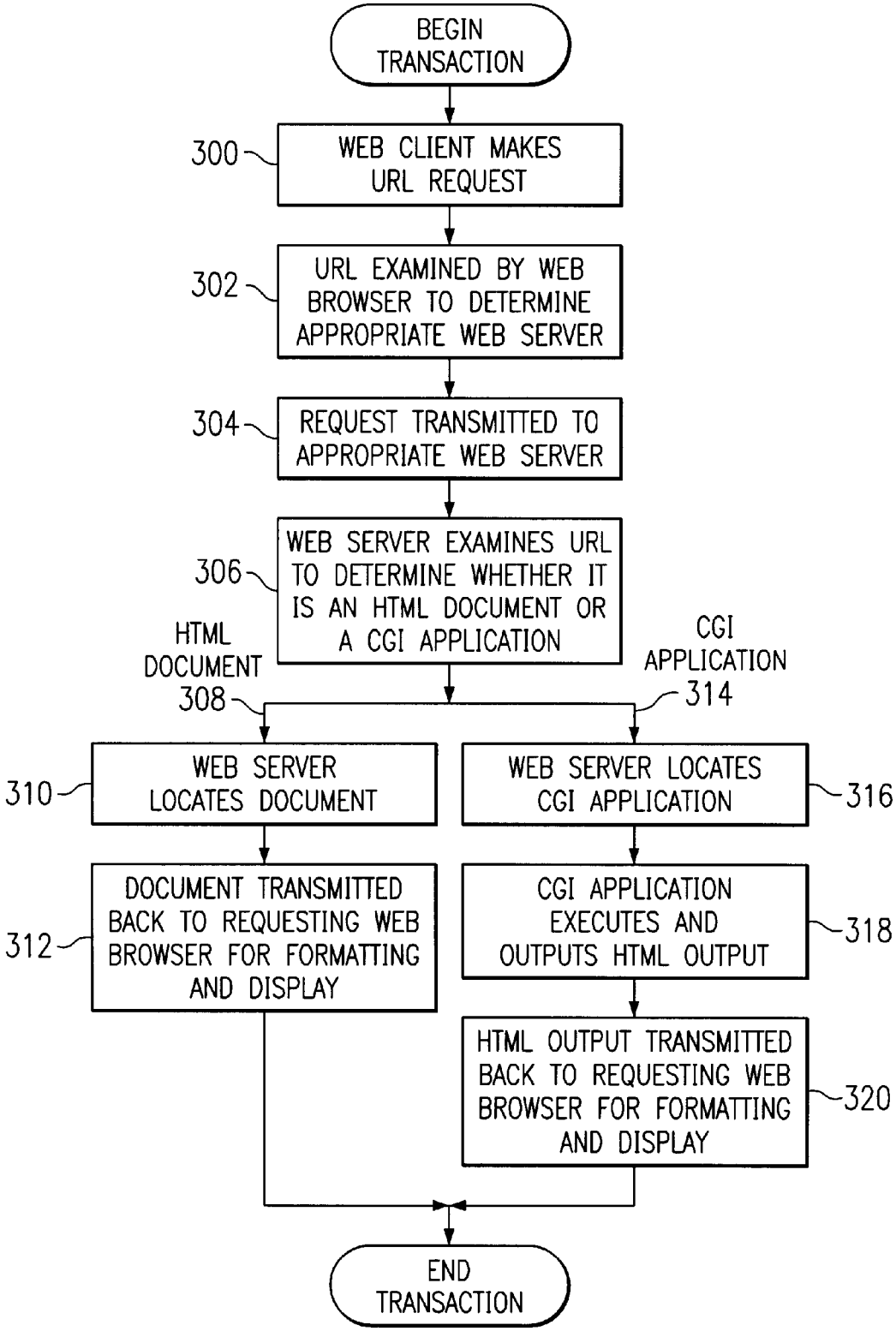


*FIG. 4*

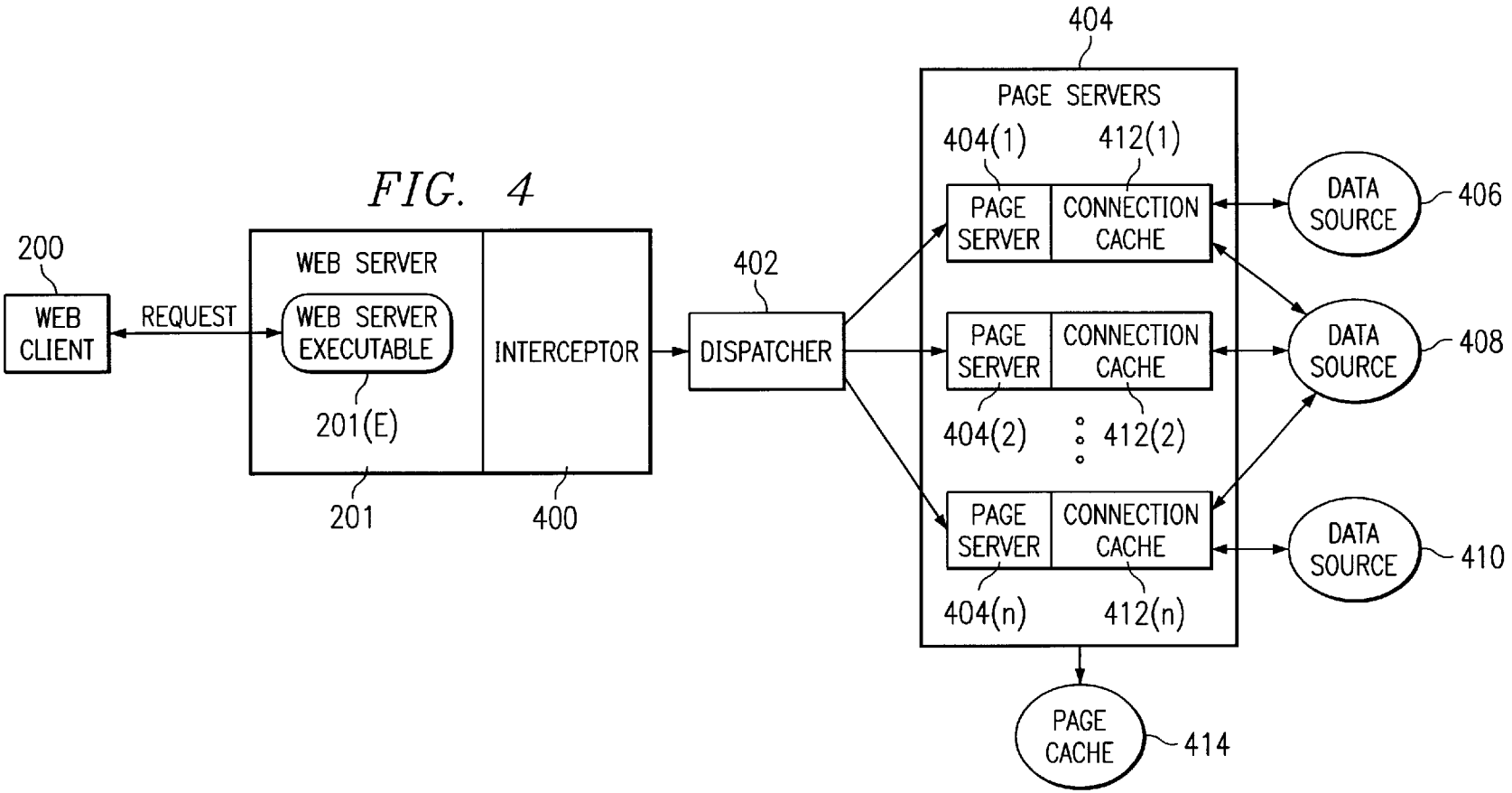
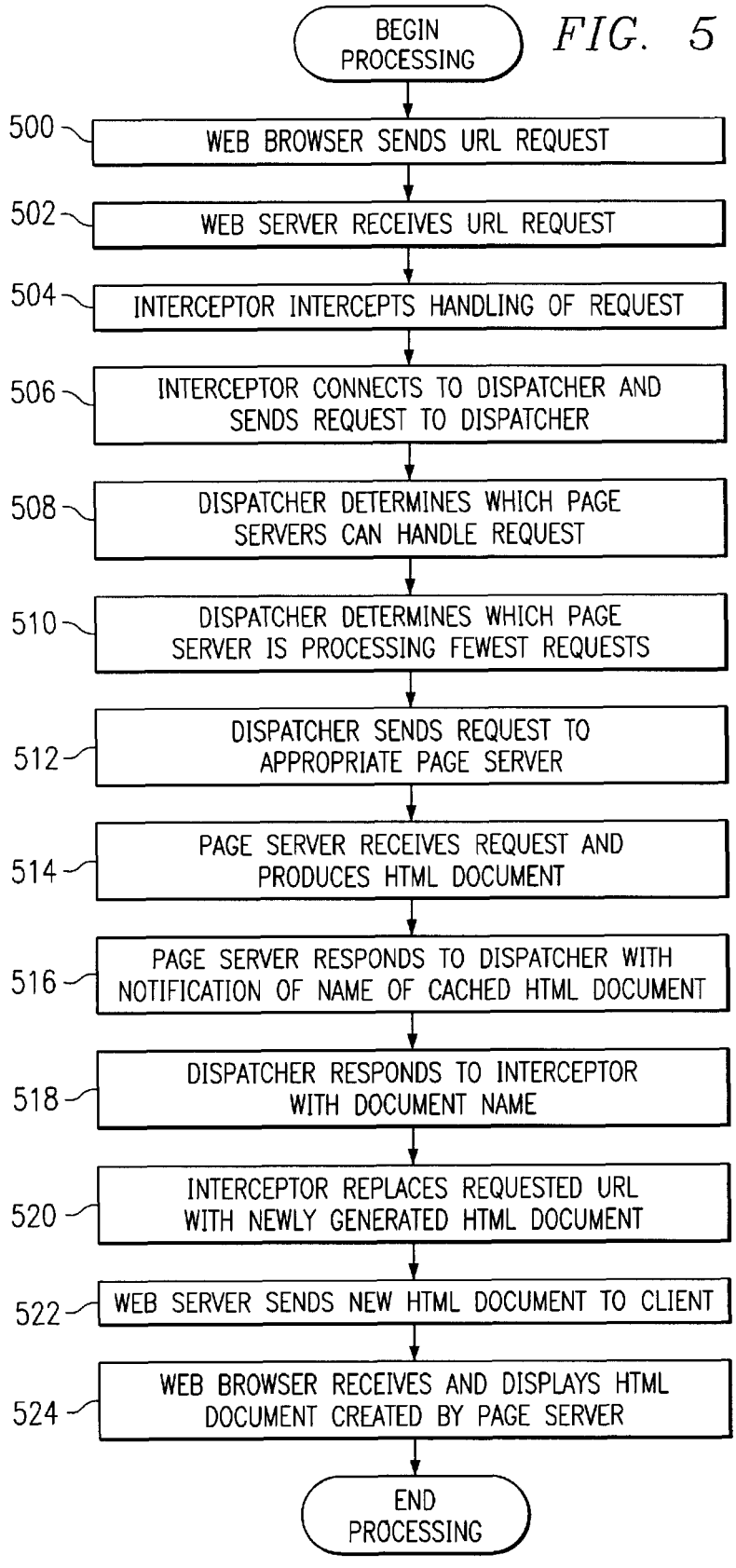

200
WEB CLIENT
REQUEST
WEB SERVER
WEB SERVER EXECUTABLE
201(E)
201
INTERCEPTOR
400
402
DISPATCHER
404
PAGE SERVERS
404(1)
412(1)
PAGE SERVER
CONNECTION CACHE
DATA SOURCE
406
PAGE SERVER
CONNECTION CACHE
DATA SOURCE
408
404(2)
412(2)
PAGE SERVER
CONNECTION CACHE
DATA SOURCE
410
404(n)
412(n)
PAGE CACHE
414

FIG. 5


BEGIN PROCESSING
500
WEB BROWSER SENDS URL REQUEST
502
WEB SERVER RECEIVES URL REQUEST
504
INTERCEPTOR INTERCEPTS HANDLING OF REQUEST
506
INTERCEPTOR CONNECTS TO DISPATCHER AND SENDS REQUEST TO DISPATCHER
508
DISPATCHER DETERMINES WHICH PAGE SERVERS CAN HANDLE REQUEST
510
DISPATCHER DETERMINES WHICH PAGE SERVER IS PROCESSING FEWEST REQUESTS
512
DISPATCHER SENDS REQUEST TO APPROPRIATE PAGE SERVER
514
PAGE SERVER RECEIVES REQUEST AND PRODUCES HTML DOCUMENT
516
PAGE SERVER RESPONDS TO DISPATCHER WITH NOTIFICATION OF NAME OF CACHED HTML DOCUMENT
518
DISPATCHER RESPONDS TO INTERCEPTOR WITH DOCUMENT NAME
520
INTERCEPTOR REPLACES REQUESTED URL WITH NEWLY GENERATED HTML DOCUMENT
522
WEB SERVER SENDS NEW HTML DOCUMENT TO CLIENT
524
WEB BROWSER RECEIVES AND DISPLAYS HTML DOCUMENT CREATED BY PAGE SERVER
END PROCESSING

# SYSTEM AND METHOD FOR MANAGING DYNAMIC WEB PAGE GENERATION REQUESTS

This application is a division of Ser. No. 08/636,477, filed Apr. 23, 1996, now U.S. Pat. No. 5,894,554.

## FIELD OF THE INVENTION

The present invention relates to the field of Internet technology. Specifically, the present invention relates to the creation and management of custom World Wide Web sites.

## DESCRIPTION OF RELATED ART

The World Wide Web (the Web) represents all of the computers on the Internet that offer users access to information on the Internet via interactive documents or Web pages. These Web pages contain hypertext links that are used to connect any combination of graphics, audio, video and text, in a non-linear, non-sequential manner. Hypertext links are created using a special software language known as HyperText Mark-Up Language (HTML).

Once created, Web pages reside on the Web, on Web servers or Web sites. A Web site can contain numerous Web pages. Web client machines running Web browsers can access these Web pages at Web sites via a communications protocol known as HyperText Transport Protocol (HTTP). Web browsers are software interfaces that run on World Wide Web clients to allow access to Web sites via a simple user interface. A Web browser allows a Web client to request a particular Web page from a Web site by specifying a Uniform Resource Locator (URL). A URL is a Web address that identifies the Web page and its location on the Web. When the appropriate Web site receives the URL, the Web page corresponding to the requested URL is located, and if required, HTML output is generated. The HTML output is then sent via HTTP to the client for formatting on the client's screen.

Although Web pages and Web sites are extremely simple to create, the proliferation of Web sites on the Internet highlighted a number of problems. The scope and ability of a Web page designer to change the content of the Web page was limited by the static nature of Web pages. Once created, a Web page remained static until it was manually modified. This in turn limited the ability of Web site managers to effectively manage their Web sites.

The Common Gateway Interface (CGI) standard was developed to resolve the problem of allowing dynamic content to be included in Web pages. CGI "calls" or procedures enable applications to generate dynamically created HTML output, thus creating Web pages with dynamic content. Once created, these CGI applications do not have to be modified in order to retrieve "new" or dynamic data. Instead, when the Web page is invoked, CGI "calls" or procedures are used to dynamically retrieve the necessary data and to generate a Web page.

CGI applications also enhanced the ability of Web site administrators to manage Web sites. Administrators no longer have to constantly update static Web pages. A number of vendors have developed tools for CGI based development, to address the issue of dynamic Web page generation. Companies like Spider™ and Bluestone™, for example, have each created development tools for CGI-based Web page development. Another company, Haht Software™, has developed a Web page generation tool that uses a BASIC-like scripting language, instead of a CGI scripting language.



Tools that generate CGI applications do not, however, resolve the problem of managing numerous Web pages and requests at a Web site. For example, a single company may maintain hundreds of Web pages at their Web site. Current Web server architecture also does not allow the Web server to efficiently manage the Web page and process Web client requests. Managing these hundreds of Web pages in a coherent manner and processing all requests for access to the Web pages is thus a difficult task. Existing development tools are limited in their capabilities to facilitate dynamic Web page generation, and do not address the issue of managing Web requests or Web sites.

## SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a method and apparatus for creating and managing custom Web sites. Specifically, the present invention claims a method and apparatus for managing dynamic web page generation requests.

In one embodiment, the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests, processing the request, the processing being performed by the page server concurrently with the Web server, as the Web server processes the other requests, and dynamically generating a Web page in response to the request, the Web page including data dynamically retrieved from one or more data sources. Other embodiments also include connection caches to the one or more data sources, page caches for each page server, and custom HTML extension templates for configuring the Web page.

Other objects, features and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a typical computer system in which the present invention operates.

FIG. **2** illustrates a typical prior art Web server environment.

FIG. **3** illustrates a typical prior art Web server environment in the form of a flow diagram.

FIG. **4** illustrates one embodiment of the presently claimed invention.

FIG. **5** illustrates the processing of a Web browser request in the form of a flow diagram, according to one embodiment of the presently claimed invention.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention relates to a method and apparatus for creating and managing custom Web sites. In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent to one of ordinary skill in the art, however, that these specific details need not be used to practice the present invention. In other instances, well-known structures, interfaces and processes have not been shown in detail in order not to unnecessarily obscure the present invention.

FIG. **1** illustrates a typical computer system **100** in which the present invention operates. The preferred embodiment of

the present invention is implemented on an IBM™ Personal Computer manufactured by IBM Corporation of Armonk, New York. An alternate embodiment may be implemented on an RS/6000™ Workstation manufactured by IBM Corporation of Armonk, New York. It will be apparent to those of ordinary skill in the art that other computer system architectures may also be employed.

In general, such computer systems as illustrated by FIG. **1** comprise a bus **101** for communicating information, a processor **102** coupled with the bus **101** for processing information, main memory **103** coupled with the bus **101** for storing information and instructions for the processor **102**, a read-only memory **104** coupled with the bus **101** for storing static information and instructions for the processor **102**, a display device **105** coupled with the bus **101** for displaying information for a computer user, an input device **106** coupled with the bus **101** for communicating information and command selections to the processor **102**, and a mass storage device **107**, such as a magnetic disk and associated disk drive, coupled with the bus **101** for storing information and instructions. A data storage medium **108** containing digital information is configured to operate with mass storage device **107** to allow processor **102** access to the digital information on data storage medium **108** via bus **101**.

Processor **102** may be any of a wide variety of general purpose processors or microprocessors such as the Pentium™ microprocessor manufactured by Intel™ Corporation or the RS/6000™ processor manufactured by IBM Corporation. It will be apparent to those of ordinary skill in the art, however, that other varieties of processors may also be used in a particular computer system. Display device **105** may be a liquid crystal device, cathode ray tube (CRT), or other suitable display device. Mass storage device **107** may be a conventional hard disk drive, floppy disk drive, CD-ROM drive, or other magnetic or optical data storage device for reading and writing information stored on a hard disk, a floppy disk, a CD-ROM a magnetic tape, or other magnetic or optical data storage medium. Data storage medium **108** may be a hard disk, a floppy disk, a CD-ROM, a magnetic tape, or other magnetic or optical data storage medium.

In general, processor **102** retrieves processing instructions and data from a data storage medium **108** using mass storage device **107** and downloads this information into random access memory **103** for execution. Processor **102**, then executes an instruction stream from random access memory **103** or read-only memory **104**. Command selections and information input at input device **106** are used to direct the flow of instructions executed by processor **102**. Equivalent input device **106** may also be a pointing device such as a conventional mouse or trackball device. The results of this processing execution are then displayed on display device **105**.

The preferred embodiment of the present invention is implemented as a software module, which may be executed on a computer system such as computer system **100** in a conventional manner. Using well known techniques, the application software of the preferred embodiment is stored on data storage medium **108** and subsequently loaded into and executed within computer system **100**. Once initiated, the software of the preferred embodiment operates in the manner described below.

FIG. **2** illustrates a typical prior art Web server environment. Web client **200** can make URL requests to Web server **201** or Web server **202**. Web servers **201** and **202** include Web server executables, **201**(E) and **202**(E) respectively, that perform the processing of Web client requests. Each Web server may have a number of Web pages **201**(1)–(n) and **202**(1)–(n). Depending on the URL specified by the Web client **200**, the request may be routed by either Web server executable **201**(E) to Web page **201** (1), for example, or from Web server executable **202**(E) to Web page **202** (1). Web client **200** can continue making URL requests to retrieve other Web pages. Web client **200** can also use hyperlinks within each Web page to "jump" to other Web pages or to other locations within the same Web page.

FIG. **3** illustrates this prior art Web server environment in the form of a flow diagram. In processing block **300**, the Web client makes a URL request. This URL request is examined by the Web browser to determine the appropriate Web server to route the request to in processing block **302**. In processing block **304** the request is then transmitted from the Web browser to the appropriate Web server, and in processing block **306** the Web server executable examines the URL to determine whether it is a HTML document or a CGI application. If the request is for an HTML document **308**, then the Web server executable locates the document in processing block **310**. The document is then transmitted back through the requesting Web browser for formatting and display in processing block **312**.

If the URL request is for a CGI application **314**, however, the Web server executable locates the CGI application in processing block **316**. The CGI application then executes and outputs HTML output in processing block **318** and finally, the HTML output is transmitted back to requesting Web browser for formatting and display in processing block **320**.

This prior art Web server environment does not, however, provide any mechanism for managing the Web requests or the Web sites. As Web sites grow, and as the number of Web clients and requests increase, Web site management becomes a crucial need.

For example, a large Web site may receive thousands of requests or "hits" in a single day. Current Web servers process each of these requests on a single machine, namely the Web server machine. Although these 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. As such, the Web executable thread may hand off a request to a processing thread, but both threads will still have to be handled by the processor on the Web server machine. When numerous requests are being simultaneously processed by multiple threads on a single machine, the Web server can slow down significantly and become highly inefficient. The claimed invention addresses this need by utilizing a partitioned architecture to facilitate the creation and management of custom Web sites and servers.

FIG. **4** illustrates one embodiment of the presently claimed invention. Web client **200** issues a URL request that is processed to determined proper routing. In this embodiment, the request is routed to Web server **201**. Instead of Web server executable **201**(E) processing the URL request, however, Interceptor **400** intercepts the request and routes it to Dispatcher **402**. In one embodiment, Interceptor **400** resides on the Web server machine as an extension to Web server **201**. This embodiment is appropriate for Web servers such as Netsite™ from Netscape, that support such extensions. A number of public domain Web servers, such as NCSA™ from the National Center for Supercomputing Applications at the University of Illinois, Urbana-Champaign, however, do not provide support for

this type of extension. Thus, in an alternate embodiment, Interceptor **400** is an independent module, connected via an "intermediate program" to Web server **201**. This intermediate program can be a simple CGI application program that connects Interceptor **400** to Web server **201**. Alternate intermediate programs the perform the same functionality can also be implemented.

In one embodiment of the invention, Dispatcher **402** resides on a different machine than Web server **201**. This embodiment overcomes the limitation described above, in prior art Web servers, wherein all processing is performed by the processor on a single machine. By routing the request to Dispatcher **402** residing on a different machine than the Web server executable **201**(E), the request can then be processed by a different processor than the Web server executable **201**(E). Web server executable **201**(E) is thus free to continue servicing client requests on Web server **201** while the request is processed "off-line," at the machine on which Dispatcher **402** resides.

Dispatcher **402** can, however, also reside on the same machine as the Web server. The Web site administrator has the option of configuring Dispatcher **402** on the same machine as Web server **201**, taking into account a variety of factors pertinent to a particular Web site, such as the size of the Web site, the number of Web pages and the number of hits at the Web site. Although this embodiment will not enjoy the advantage described above, namely off-loading the processing of Web requests from the Web server machine, the embodiment does allow flexibility for a small Web site to grow. For example, a small Web site administrator can use a single machine for both Dispatcher **402** and Web server **201** initially, then off-load Dispatcher **402** onto a separate machine as the Web site grows. The Web site can thus take advantage of other features of the present invention regardless of whether the site has separate machines configured as Web servers and dispatchers.

Dispatcher **402** receives the intercepted request and then dispatches the request to one of a number of Page servers **404 (1)**–(n). For example, if Page server **404 (1)** receives the dispatched request, it processes the request and retrieves the data from an appropriate data source, such as data source **406**, data source **408**, or data source **410**. Data sources, as used in the present application, include databases, spreadsheets, files and any other type of data repository. Page server **404 (1)** can retrieve data from more than one data source and incorporate the data from these multiple data sources in a single Web page.

In one embodiment, each Page server **404(1)**–(n) resides on a separate machine on the network to distribute the processing of the request. Dispatcher **402** maintains a variety of information regarding each Page server on the network, and dispatches requests based on this information. For example, Dispatcher **402** retains dynamic information regarding the data sources that any given Page server can access. Dispatcher **402** thus examines a particular request and determines which Page servers can service the URL request. Dispatcher **402** then hands off the request to the appropriate Page server.

For example, if the URL request requires financial data from data source **408**, dispatcher **402** will first examine an information list. Dispatcher **402** may determine that Page server **404(3)**, for example, has access to the requisite data in data source **408**. Dispatcher **402** will thus route the URL request to Page server **404(3)**. This "connection caching" functionality is described in more detail below, under the heading "Performance." Alternately, Dispatcher **402** also has the ability to determine whether a particular Page server already has the necessary data cached in the Page server's page cache (described in more detail below, under the heading "Performance"). Dispatcher **402** may thus determine that Page server **404(1)** and **404(2)** are both logged into Data source **408**, but that Page server **404(2)** has the financial information already cached in Page server **404(2)**'s page cache. In this case, Dispatcher **402** will route the URL request to Page server **404(2)** to more efficiently process the request.

Finally, Dispatcher **402** may determine that a number or all Page servers **404(1)**–(n) are logged into Data source **408**. In this scenario, Dispatcher **402** can examine the number of requests that each Page server is servicing and route the request to the least busy page server. This "load balancing" capability can significantly increase performance at a busy Web site and is discussed in more detail below, under the heading "Scalability".

If, for example, Page server **404(2)**, receives the request, Page server **404(2)** will process the request. While Page server **404(2)** is processing the request, Web server executable **201**(E) can concurrently process other Web client requests. This partitioned architecture thus allows both Page server **404(2)** and Web server executable **201**(E) to. simultaneously process different requests, thus increasing the efficiency of the Web site. Page server **404(2)** dynamically generates a Web page in response to the Web client request, and the dynamic Web page is then either transmitted back to requesting Web client **200** or stored on a machine that is accessible to Web server **201**, for later retrieval.

One embodiment of the claimed invention also provides a Web page designer with HTML extensions, or "dyna" tags. These dyna tags provide customized HTML functionality to a Web page designer, to allow the designer to build customized HTML templates that specify the source and placement of retrieved data. For example, in one embodiment, a "dynatext" HTML extension tag specifies a data source and a column name to allow the HTML template to identify the data source to log into and the column name from which to retrieve data. Alternatively, "dyna-anchor" tags allow the designer to build hyperlink queries while "dynablock" tags provide the designer with the ability to iterate through blocks of data. Page servers use these HTML templates to create dynamic Web pages. Then, as described above, these dynamic Web pages are either transmitted back to requesting Web client **200** or stored on a machine that is accessible to Web server **201**, for later retrieval.

The presently claimed invention provides numerous advantages over prior art Web servers, including advantages in the areas of performance, security, extensibility and scalability

## Performance

One embodiment of the claimed invention utilizes connection caching and page caching to improve performance. Each Page server can be configured to maintain a cache of connections to numerous data sources. For example, as illustrated in FIG. **4**, Page server **404(1)** can retrieve data from data source **406**, data source **408** or data source **410**. Page server **404(1)** can maintain connection cache **412(1)**, containing connections to each of data source **406**, data source **408** and data source **410**, thus eliminating connect times from the Page servers to those data sources.

Additionally, another embodiment of the present invention supports the caching of finished Web pages, to optimize the performance of the data source being utilized. This "page

caching" feature, illustrated in FIG. **4** as Page cache **414**, allows the Web site administrator to optimize the performance of data sources by caching Web pages that are repeatedly accessed. Once the Web page is cached, subsequent requests or "hits" will utilize the cached Web page rather than re-accessing the data source. This can radically improve the performance of the data source.

### Security

The present invention allows the Web site administrator to utilize multiple levels of security to manage the Web site. In one embodiment, the Page server can utilize all standard encryption and site security features provided by the Web server. In another embodiment, the Page server can be configured to bypass connection caches **412(1)**–(n), described above, for a particular data source and to require entry of a user-supplied identification and password for the particular data source the user is trying to access.

Additionally, another embodiment of the presently claimed invention requires no real-time access of data sources. The Web page caching ability, described above, enables additional security for those sites that want to publish non-interactive content from internal information systems, but do not want real-time Internet accessibility to those internal information systems. In this instance, the Page server can act as a "replication and staging agent" and create Web pages in batches, rather than in real-time. These "replicated" Web pages are then "staged" for access at a later time, and access to the Web pages in this scenario is possible even if the Page server and dispatcher are not present later.

In yet another embodiment, the Page server can make a single pass through a Web library, and compile a Web site that exists in the traditional form of separately available files. A Web library is a collection of related Web books and Web pages. More specifically, the Web library is a hierarchical organization of Web document templates, together with all the associated data source information. Information about an entire Web site is thus contained in a single physical file, thus simplifying the problem of deploying Web sites across multiple Page servers. The process of deploying the Web site in this embodiment is essentially a simple copy of a single file.

### Extensibility

One embodiment of the present invention provides the Web site administrator with Object Linking and Embedding (OLE) 2.0 extensions to extend the page creation process. These OLE 2.0 extensions also allow information submitted over the Web to be processed with user-supplied functionality. Utilizing development tools such as Visual Basic, Visual C++ or PowerBuilder that support the creation of OLE 2.0 automation, the Web site administrator can add features and modify the behavior of the Page servers described above. This extensibility allows one embodiment of the claimed invention to be incorporated with existing technology to develop an infinite number of custom web servers.

For example, OLE 2.0 extensions allow a Web site administrator to encapsulate existing business rules in an OLE 2.0 automation interface, to be accessed over the Web. One example of a business rule is the steps involved in the payoff on an installment or mortgage loan. The payoff may involve, for example, taking into account the current balance, the date and the interest accrued since the last payment. Most organizations already have this type of business rule implemented using various applications, such as Visual Basic for client-server environments, or CICS programs on mainframes. If these applications are OLE 2.0 compliant, the Page server "dynaobject" HTML extension tag can be used to encapsulated the application in an OLE 2.0 automation interface. The Page server is thus extensible, and can incorporate the existing application with the new Page server functionality.

### Scalability

One embodiment of the claimed invention allows "plug and play" scalability. As described above, referring to FIG. **4**, Dispatcher **402** maintains information about all the Page servers configured to be serviced by Dispatcher **402**. Any number of Page servers can thus be "plugged" into the configuration illustrated in FIG. **4**, and the Page servers will be instantly activated as the information is dynamically updated in Dispatcher **402**. The Web site administrator can thus manage the overhead of each Page server and modify each Page server's load, as necessary, to improve performance. In this manner, each Page server will cooperate with other Page servers within a multi-server environment. Dispatcher **402** can examine the load on each Page server and route new requests according to each Page server's available resources. This "load-balancing" across multiple Page servers can significantly increase a Web site's performance.

FIG. **5** illustrates the processing of a Web browser request in the form of a flow diagram, according to one embodiment of the presently claimed invention. A Web browser sends a URL request to a Web server in processing block **500**. In processing block **502**, the Web server receives the URL request, and an interceptor then intercepts the handling of the request in processing block **504**. The interceptor connects to a dispatcher and sends the URL request to the dispatcher in processing block **506**. In processing block **508**, the dispatcher determines which Page servers can handle the request. The dispatcher also determines which Page server is processing the fewest requests in processing block **510**, and in processing block **512**, the dispatcher sends the URL request to an appropriate Page server. The Page server receives the request and produces an HTML document in processing block **514**. The Page server then responds to the dispatcher with notification of the name of the cached HTML document in processing block **516**. In processing block **518**, the dispatcher responds to the interceptor with the document name, and the interceptor then replaces the requested URL with the newly generated HTML document in processing block **520**. The Web server then sends the new HTML document to the requesting client in processing block **522**. Finally, the Web browser receives and displays the HTML document created by the Page server at processing block **524**.

Thus, a method and apparatus for creating and managing custom Web sites is disclosed. These specific arrangements and methods described herein are merely illustrative of the principles of the present invention. Numerous modifications in form and detail may be made by those of ordinary skill in the art without departing from the scope of the present invention. Although this invention has been shown in relation to a particular preferred embodiment, it should not be considered so limited. Rather, the present invention is limited only by the scope of the appended claims.

We claim:

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

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

Web server to process other requests wherein said routing step further includes the steps of:

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

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

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

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

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

dispatching said request to said page server.

**3**. The computer-implemented method in claim **1** wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data.

**4**. The computer-implemented method in claim **1** wherein said step of dynamically generating said Web page includes the step of dynamically retrieving said data from said one or more data sources.

**5**. The computer-implemented method in claim **1** wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources.

**6**. The computer-implemented method in claim **1** wherein said step of processing said request includes the step of logging into said one or more data sources.

**7**. The computer-implemented method in claim **1** wherein said step of dynamically generating said Web page includes the step of maintaining a page cache containing said Web page.

**8**. The computer-implemented method in claim **1** wherein said page server includes tag-based text templates for configuring said Web page.

**9**. The computer-implemented method in claim **8** wherein said step of processing said request further includes the step of inserting said-dynamically retrieved data from said one or more data sources into said tag-based text templates.

**10**. The computer-implemented method in claim **8** wherein at least one of said tag-based text templates drives a format of the data dynamically retrieved from said one or more data sources in response to said request.

**11**. The computer-implemented method in claim **8** wherein said tag-based text templates include HTML templates.

**12**. The computer-implemented method in claim **1** wherein said step of processing said request further includes the step of dynamically updating data at said one or more data sources.

**13**. The computer-implemented method in claim **1** wherein said step of processing said request further includes the step of processing an object handling extension.

**14**. The computer-implemented method in claim **13** wherein said object handling extension is an OLE extension.

**15**. A computer-implemented method comprising the steps of:

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

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

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

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

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

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

dispatching said request to said page server.

**17**. The computer-implemented method in claim **15** wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data.

**18**. The computer-implemented method in claim **15** wherein said step of dynamically generating said page includes the step of dynamically retrieving said data from said one or more data sources.

**19**. The computer-implemented method in claim **15** wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources.

**20**. The computer-implemented method in claim **15** wherein said step of processing said request includes the step of logging into said one or more data sources.

**21**. The computer-implemented method in claim **15** wherein said step of dynamically generating said page includes the step of maintaining a page cache containing said page.

**22**. The computer-implemented method in claim **15** wherein said page server includes tag-based text templates for configuring said page.

**23**. The computer-implemented method in claim **22** wherein said step of processing said request further includes the step of inserting said dynamically retrieved data from said one or more data sources into said tag-based text templates.

**24**. The computer-implemented method in claim **22** wherein at least one of said tag-based text templates drives a format of the data dynamically retrieved from said one or more data sources in response to said request.

**25**. The computer-implemented method in claim **22** wherein said tag-based text templates include HTML templates.

**26**. The computer-implemented method in claim **15** wherein said step of processing said request further includes the step of dynamically updating data at said one or more data sources.

**27**. The computer-implemented method in claim **15** wherein said step of processing said request further includes the step of processing an object handling extension.

**28**. The computer-implemented method in claim **27** wherein said object handling extension is an OLE extension.

**29**. A computer-implemented method comprising the steps of:

transferring a request from an HTTP-compliant device to a dispatcher;

maintaining dynamic information regarding data sources a given page server may access;

dispatching said request to an appropriate page server based on said request and based on said dynamic information, said page server receiving said request and

releasing said HTTP-compliant device to process other requests;

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



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

* * * * *

# Exhibit 3

**IN THE UNITED STATES DISTRICT COURT**
**FOR THE DISTRICT OF DELAWARE**

| | |
|---|---|
| **ORACLE CORPORATION and ORACLE U.S.A. INC.,**<br><br>**Plaintiffs,**<br><br>**v.**<br><br>**EPICREALM LICENSING, LLC,**<br><br>**Defendant.** | **C.A. No. 06-cv-414 (SLR)** |
| **AND RELATED COUNTERCLAIMS** | |

**REPORT OF PLAINTIFFS' EXPERT MICHAEL I. SHAMOS, PH.D, J.D. CONCERNING INVALIDITY**

**BACKGROUND & QUALIFICATIONS**

1. My name is Michael I. Shamos. I hold the title of Distinguished Career Professor in the School of Computer Science at Carnegie Mellon University in Pittsburgh, Pennsylvania. I was a founder and Co-Director of the Institute for eCommerce at Carnegie Mellon and I now direct a graduate degree program in eBusiness Technologies. My résumé is attached as Exhibit 2 to this report.

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

g. The person of ordinary skill is a hypothetical person who is presumed to be aware of all of the pertinent art. The person of ordinary skill is not an automaton, and may be able to fit together the teachings of multiple patents employing ordinary creativity and the common sense that familiar items may have obvious uses beyond their primary purposes. It is not necessary to demonstrate precise teachings directed to the specific subject matter of the challenged claim, for a court can take account of the inferences and creative steps that a person of ordinary skill in the art would employ. A patent which merely claims predictable uses of old elements according to their established functions to achieve predicable results should be found invalid as obvious.

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

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

**THE SUBJECT MATTER**

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

26. After the Internet was made available for general commercial use in the mid 1990s, the World Wide Web expanded prodigiously. Web browsing consists primarily in the viewing of Web pages, which are transmitted over the Internet from Web servers to individual machines called Web clients which run Web browsers. The principal function of a Web browser is to display, or "render" Web pages so they can be viewed by humans on display screens.

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

company's products. Static pages can be delivered quickly because virtually no processing is required to transmit them over a communication line.

28. Static pages, of course, are not suitable for information which changes frequently, such as the price of a stock, or for information that is retrieved or computed on a custom basis at the moment it is requested. Such pages are called "dynamic pages" because they are created dynamically on request. As a general matter, dynamic Web pages by their nature cannot be created or stored in advance[1]. A good example of a dynamic page is an individual user's shopping cart at a retail website. Obviously the shopping cart cannot be prestored because it is created dynamically in response to the shopping activity of the user. Another example of a dynamic page is one produced by a search engine such as Google, Lycos or Yahoo! in response to a query. Not only is a search engine's index of web pages changing dynamically as pages are being added to the Web and indexed by the engine, but there is no possible way to anticipate every possible query that might be posed. Therefore, the search engine response pages cannot be static, but must be generated on demand.

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

30. In this report, the phrase "it was known" means throughout that "it was known prior to the inventions claimed in the Patents."

[1] It is possible to store, or "cache" a dynamic page if there is an expectation that the very same page will be requested multiple times. Such caching would reduce the time necessary to generate the page repeatedly. However, it is not possible to store all possible dynamic pages that might be requested.

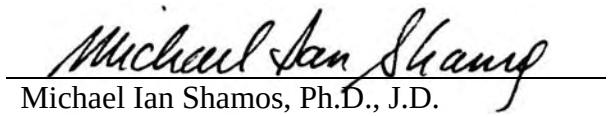
823. Applicants did not invent object handling extensions.

824. Applicants did not invent Object Linking and Embedding (OLE).

825. Applicants did not invent HTTP-compliant devices.

826. Applicants did not invent any of the claimed combinations of the foregoing to respond to dynamic page generation requests.

Executed on May 2, 2008, in Pittsburgh, PA.



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

# Exhibit 4

# Oracle® WebServer User's Guide

Release 1.0
Part No. A34986–2

ORACLE®

MAC003037

Oracle WebServer User's Guide, Release 1.0

Part No. A34986–2

Copyright © 1995 Oracle Corporation

All rights reserved. Printed in the U.S.A.

Contributing Authors: Mala Anand, Matthew Bookman, Martin Gruber, Magnus Lonnroth,Katrina Montinola, Mary Owen, and Joe Pistritto

Contributors: Arthur Gleckler, Walter Hudson, Kelly Ireland, Aileen Jaitin, Max Pollak, Sandy Venning, and Marcel Weekes

This software was not developed for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It is the customer's responsibility to take all appropriate measures to ensure the safe use of such applications if the programs are used for such purposes.

This software/ documentation contains proprietary information of Oracle Corporation; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited.

If this software/ documentation is delivered to a U.S. Government Agency of the Department of Defense, then it is delivered with Restricted Rights and the following legend is applicable:

Restricted Rights Legend Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of DFARS 252.227–7013, Rights in Technical Data and Computer Software (October 1988).

Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

If this software/ documentation is delivered to a U.S. Government Agency not within the Department of Defense, then it is delivered with " Restricted Rights", as defined in FAR 52.227–14, Rights in Data – General, including Alternate III (June 1987).

The information in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free.

Oracle, SQL*Forms, SQL*DBA, SQL*Loader, SQL*Net and SQL*Plus are registered trademarks of Oracle Corporation.

PL/ SQL and Oracle7 are trademarks of Oracle Corporation.

All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.

CHAPTER

# 3 Oracle Web Listener

The Oracle Web Listener is a high–performance scalable and portable HTTP server, supporting high–traffic Web applications where quick response is crucial. The Oracle Web Listener is designed to enable global distributed information systems over a corporate network or the global Internet.

From a user's perspective, the Web is a collection of documents or pages which contain text, images, and hypertext links to other pages. By pointing, clicking, and traversing the links, the user has instant access to a distributed collection of information. The Oracle Web Listener merges the techniques of information retrieval and hypertext to create a powerful global information system.

The Oracle Web Listener works on a simple client–server model. Clients send requests to the Oracle Web Listener. The listener interprets a request by reading the URL, finds (or generates) the information requested, and returns that information to the client in the form requested.

You can run several Oracle Web Listeners on the same machine by giving each a dedicated port. Having multiple Web Listeners is a good way to balance the load among various applications.

The language that the Oracle Web Listener uses to communicate with its clients is HyperText Transfer Protocol (HTTP). The Web Listener implements version 1.0 of the HTTP protocol as described in the IETF HTTP Working Group (http:/ / www.ics.uci.edu/ pub/ ietf/ http/ ). All



MAC003085

ORCL00941761

Web clients must be able to speak the HTTP protocol in order to send and receive hypermedia documents.

HTTP is an application level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. It is a generic, stateless, object–oriented protocol which can be used for many systems, through extension of its request methods (commands).

## Oracle Web Listener Features

The Oracle Web Listener includes a number of features to enhance performance and extend its usability in a variety of environments.

### High Performance Architecture

For maximum performance the Web Listener is designed to run as an asynchronous engine via a single process with a single thread, in comparison to most HTTP servers, which start a new thread or process whenever a new connection is made. This considerably decreases the time required to service a request and reduces Listener machine resource utilization, providing high performance under a heavy load.

### File Caching in Memory

The Oracle Web Listener allows a configurable set of commonly accessed files to be cached in memory. This provides very good performance when these files are accessed by a client.

In the current version of the Oracle Web Listener, the list of files to be cached in memory is determined by the WebServer Administrator and configured manually. The administrator should periodically analyze the Listener logs to determine which files should be cached and to change the configuration accordingly.

### Memory Mapping of Files

On operating systems which support this feature, the Oracle Web Listener will automatically map files into memory addresses when they are accessed ( unless it is otherwise specified that they are to be cached permanently into memory),. In this case the file will appear to be loaded in memory, allowing more than one connection accessing the same file to avoid duplicate disk reads. In addition, the operating system will



MAC003086

ORCL00941762

usually pre–fetch the next segment of a memory–mapped file while the previous one is being transmitted, further increasing performance, even in the single connection case.

## Directory Mapping

The Oracle Web Listener supports mapping pathnames that appear externally in URLs on the Web to locations internally on the Web Listener machine. This allows a consistent appearance to the clients, even if the Web Listener's disk is reorganized.

The specific mapping of externally visible (virtual) directories to local physical directories in the file system is configurable by the WebServer Administrator.

For example, the URL

http://www.acme.com/products/info.html

can be mapped into the directory /disk1/products of system www.oracle.comas:

/disk1/products/info.html

If the products directory is moved to /disk3 in a reorganization, the Administrator reconfigures the virtual directory products to point to the new location /disk3/products in the Web Listener configuration file (owl.cfg), and the Web client is completely unaware of the change.

For a full description of the use of directory mappings to control the use of "virtual" parent directories and subdirectories, see the section "Directory Mappings" in Chapter 7, "The Oracle WebServer Administration Utility".

> Note: The Web Listener needs to be reloaded to recognize the change. For information on reloading the Web Listener, see "Web Listener Control Utility" later in this chapter.

## Language Extensions

The Oracle Web Listener supports files stored in different character sets appropriate to the language being used. The Web Listener can interpret the file extension to provide both the type of the data in the file (such as HTML data) and the national language in which the file is written. The language mapping feature is controlled by the Administrator, and allows files written in different character sets appropriate to national languages to be accessed by clients of the Web Listener.



MAC003087

ORCL00941763

The mapping of specific file extensions to language and data types is controlled by the Web Listener configuration file.

For more information about the Web Listener configuration file, see "Configuration Parameters," later in this chapter.

## HTTP Protocol Negotiation Features

One of the features of HTTP is the ability of clients and HTTP servers to exchange information about the types of data they can transfer and interpret. These negotiation features allow the construction of systems composed of different data types and languages.

When a Web client requests an object, it can specify what types of data it can receive and express a preference for specific types. The Web Listener can use the information to send the appropriately formatted data to the client if more than one format is stored on the Web Listener. This process is called HTTP type negotiation and is fully supported by the Oracle Web Listener. For example, a document might contain an inline image, which could be transferred in either GIF or JPEG format to a Web client.

The Oracle Web Listener also supports a similar concept for negotiation of the appropriate national language to display a document for a specific user. The document could be stored in multiple national languages (for instance English and Canadian French), and the appropriate copy of the document would be sent if the client expressed a preference. This makes the maintenance of a multilingual Web site a great deal easier, since there is no need to maintain duplicate URLs and separate directory trees for different languages.

## Common Gateway Interface (CGI)

Common Gateway Interface (CGI) version 1.1 is the standard technique used by an HTTP server to execute a program that generates HTML output. This technique provides dynamic content rather than static content from files on disk. The Oracle Web Listener is fully compliant with CGI 1.1.

When a request comes in that the Web Listener recognizes as a request to execute a CGI application, a separate process is created to perform the operation. The Web Listener maintains communication with the process to transmit input from the Web client to it, and to retrieve the HTML output it may generate for display on the client's screen.

The Web Listener Administrator can configure specific directories to contain CGI applications to be run instead of files to be transmitted. Either of these can be done using the Web Listener configuration file. For



MAC003088

ORCL00941764

more information on the Web Listener configuration file, see "Configuration Parameters," later in this chapter.

## CGI Application Execution

When a client requests a URL from the Oracle Web Listener, the URL's pathname component is analyzed to determine if it represents a directory containing files (the usual case). These files are then sent immediately to the client or to a directory containing CGI applications, sometimes referred to as scripts, which must be executed as separate programs to generate output which is then sent to the client.

The technique of CGI application execution is one of the key features of the Web. It allows the integration of many information sources. The Oracle Web Agent is one CGI application that can be accessed using this technique.

Once the Listener has determined that a URL represents a CGI application, it interprets the URL to extract path information and arguments to be passed to the CGI application on startup.

URLs to CGI programs are split into three different parts:

- virtual path
- extra path information
- query string

The syntax is as follows:

*virtual_path extra_path_information?query_string*

| | |
|---|---|
| virtual path | similar to a path you would use to access a regular document or image. That is, it points the server to the file that contains the CGI program you want executed. |
| extra path information | additional information you can embed in the URL after the program name. Extra path information is optional. It can be used to convey constant information to your scripts independent of the client's intervention. It can also be used to access the server's virtual–to–physical path translation mechanism. |
| query string | another optional part of the URL. It can either be explicitly given in the hypertext anchor, it can come from a user typing into a search dialog box for an HTML document with the ISINDEX tag, or it can come from HTML forms. |



MAC003089

ORCL00941765

User input from URLs and other resources

When data is typed into a search dialog or text entry form, the data is encoded using URL encoding. In this encoding the following rules apply:

- Spaces are changed into + signs
- Any of the characters can be "escaped" by changing them into a sequence of the form %xx, where x is a hexadecimal digit. The character is identified by translating the two hexadecimal digits into a number from 0–255, which represents a character.

If the data is coming from a search dialog resulting from an ISINDEX tag, the above translations can be applied directly. Further, your CGI program can receive this information fully translated on the command line if you want to avoid performing the translation yourself.

If the data is coming from an HTML form, then the location of this data varies depending on the method attribute specified with the FORM tag in your HTML document. If the GET method is used, this information comes from a QUERY_STRING variable. If the POST method is used, this information is sent to your program using standard input.

The data provided will be in the form:

*name1=val ue1&name2=val ue2. . . . . &nameN=val ueN*

If there are any equal signs (=) or ampersands (&) in the encoded data, they are encoded using the above encoding rules. To decode the data appropriately, the NAME=VALUE pairs should be split (eliminating the ampersands), then each pair should be split into a name and a value, and then the URL decoding should be applied to each portion of the pair.

When a form is submitted you can often use the order in which the form items appear to determine the order in which your CGI program receives the NAME=VALUE pairs. However, you should not depend on this behavior. The various form elements have their own ways to determine what value will be associated with the name they are given. All of the textual input areas use the user's typed input as the value.

- Radio buttons use the value of the enabled button.
- Checkboxes are unchecked. Hence, they will use either an empty string, or their name will not appear in the encoded form data at all.
- Hidden form elements can be used to send constant or state information to your script.



MAC003090

ORCL00941766

## Imagemap Support

The imagemap feature of the Oracle Web Listener allows a displayed image sent to the client to have multiple link areas that provide different hypermedia links (URLs) based on where the user clicks on the image. An example is a map of the world that allows the user to select a country by clicking on it. Use of imagemaps is extremely common in sophisticated Web sites.

The Oracle Web Listener includes native processing of imagemaps within the Web Listener. Many other HTTP servers require a separate process to be created to interpret an imagemap. By doing this internally, the Oracle Web Listener improves performance and lowers system resource utilization when this common feature of HTML is being used.

## Domain Name Server (DNS) Resolution

When a client requests a URL from an HTTP server, the client's network address (IP address) is logged by the HTTP server and can be used to control access to documents on the HTTP server. The Domain Name System (DNS) converts the client's network address to the client's hostname as an ASCII string.

> Note: The DNS process can be time consuming, since another machine where the naming system is already running may be contacted.

The Oracle Web Listener provides three options for controlling DNS resolution of network addresses:

- Always resolve addresses on connections.
- Do "lazy" resolutions where the name is not resolved until an external application, a security module, or a CGI script needs it.
- Never resolve addresses even if a client asks for it.

You can use the Oracle WebServer Administration Utility or edit the configuration file to control which type of DNS resolution is used. The default is never to resolve using DNS, which provides the best performance.

When DNS resolution does occur, the results are cached. Since there are often several connections from the same client in close succession, this helps to minimize the performance penalty if DNS resolution is required.



MAC003091

ORCL00941767

## Client to Web Listener Security

The security framework for the Oracle Web Listener includes support for Basic Authentication and Digest Authentication. Both of these schemes provide for username and password authentication on documents. Access may also be controlled based on the client's network address. Use the Oracle WebServer Administration Utility, described in Chapter 7, to configure the security features.

**Basic Authentication**

This scheme uses usernames and passwords which are sent in the clear across the network. This technique has largely been replaced in practice by Digest Authentication and is included for compatibility with older Web browsers that don't implement Digest authentication.

**Digest Authentication**

This scheme is similar to Basic Authentication, except that it uses usernames and passwords which are sent encrypted across the network. A cryptographic checksum (digest) is used to encode the password, preventing an attacker from gaining access to the original username/ password combination. This technique is significantly more secure than Basic Authentication and is recommended for all new applications, although older browsers, some of which are still quite popular, do not support Digest Authentication.

**Internet Protocol–Based Restriction**

The network address of the client may be used to restrict access to information on the Oracle Web Listener. Addresses may be individually restricted or allowed access, or entire IP networks may be specified to allow users within certain locations access while denying all others.

When a client requests a file protected by IP–based security, its address is compared against the specified addresses in the order given, and as soon as a match is found, the client is allowed or denied access as specified. Files protected by IP–based restriction may not be accessed by clients other than those listed in the configuration parameters.

Note that Digest Authentication is a safer technique to determine who a user really is, since a credential (the password) must be presented to gain access. Network address–based restriction techniques are very convenient in that they avoid password management problems, but are also less secure, since a clever attacker may falsify network addresses.

**Domain Name –Based Restriction**

Domain name–based restriction is similar to IP–based restriction. A list of groups and hostnames specifies which hosts should or should not have access to a set of documents.

This technique uses symbolic hostnames rather than network addresses. Hence, it is easier to administer since network addresses may change if the network architecture is changed. This technique still suffers from the



MAC003092

ORCL00941768

problem of clever attackers falsifying their network addresses and assuming the identity of a host other than their own.

**Security on Specific Files**

Specific files can be made secure by associating them with Basic Authentication, Digest Authentication, IP–based restriction, domain–name based restriction, or any combination of one authentication and one restriction method.

**Other Security Features**

In addition to the authentication features, the Web Listener's user and group ID (on UNIX implementations) may be specified in the configuration file so that the Web Listener itself can be restricted to accessing only files available to that user and group. This allows the Web Listener to be started as a privileged user as required on some implementations, but still operates as if it were an unprivileged program. This provides additional security against an attack directly on the Web Listener process. For more information on the use of the Group ID for security purposes, see "Basic Configuration Parameters" in Chapter 7 of this document.

## Logging

When a request is made of the Web Listener, the request information may be logged to a specified log file. The Oracle Web Listener records connections in the industry standard Common Log Format. This format includes the first line of the client's request, which specifies the action, URL, and protocol version. It also includes such information as the client's host address, whether the request was successful, and how many bytes were transmitted. This information may be later used to analyze what types of accesses are performed by which users, and to check for errors. Use the WebServer Administration Utility to specify the log file directory and the names of the standard and error log files in the Web Listener configuration file.

> Note: Be sure that the user ID under which the Web Listener runs has privileges to read and write to the log file directory.

## Starting Up and Shutting Down the Oracle Web Listener

The easiest way to start up, shut down, or reload an Oracle Web Listener is to use the Administration Utility described in Chapter 7 of this document. If you wish to perform these operations from the UNIX operating system, use the WLCTL Utility as described in the next section.



MAC003093

ORCL00941769

## The WLCTL (Web Listener Control) Utility

This utility resides in the ORACLE_HOME bin directory and allows you to easily startup, stop, and reload your Oracle Web Listener.

The syntax you use to invoke WLCTL to perform various operations is as follows:

```
> wlctl [start|stop|reload] [port_number] ORACLE_HOME
```

If ORACLE_HOME is set in your environment and is the ORACLE_HOME where the Oracle WebServer is installed, it need not be included on the command line.

The following examples assume that the Oracle WebServer is located in /u01/oracle:

- With ORACLE_HOME set:

  ```
  >wlctl start 8888
  ```

- Specifying ORACLE_HOME on the command line:

  ```
  >wlctl start 8888 /u01/oracle
  ```

- With ORACLE_HOME set:

  ```
  wlctl stop 8888
  ```

- Specifying ORACLE_HOME on the command line:

  ```
  wlctl stop 8888 /u01/oracle
  ```

- With ORACLE_HOME set:

  ```
  >wlctl reload 8888
  ```

- Specifying ORACLE_HOME on the command line:

  ```
  >wlctl reload 8888 /u01/oracle
  ```

The stop and reload functions require that the Listener PID file for the Web Listener you are stopping or reloading resides in the location pointed to by the Web Listener configuration file.

## Oracle Web Listener Configuration Parameters

The configuration parameters for the Oracle Web Listener are stored in owa.cfg, the Web Listener configuration file. It is read when the Web Listener is initially started and on receipt of a signal '1' on UNIX implementations. The Web Listener configuration file is specified with the –c option on the command line when the Web Listener is started,



MAC003094

ORCL00941770

allowing multiple Web Listeners with different configuration files to be started on the same Web Listener machine.

Oracle WebServer provides an HTML–based Administration Utility which may be accessed with any forms–capable Web browser. This utility eliminates the need for the WebServer administrator to edit the Web Listener configuration file manually in most cases, and includes explanatory help text on the individual parameters. The Administration Utility is described in Chapter 7, "The Oracle WebServer Administration Utility". This section documents the parameters in the configuration file for completeness, in the event an administrator wishes to edit the file manually.

The Web Listener configuration file is divided into sections, which start with a section name in brackets—for example, [NetInfo]. Individual configuration parameters are set by name = value pairs, with the configuration parameter on the left of the equal sign and the value on the right as specified in the table below.

For example, consider the following portion of a configuration file:

```
;
;  www.acme.com configuration file
;
[NetInfo]
HostName  =  www.acme.com
HostAddress  =  ANY
PortNumber  =  80
```

This portion of the file sets HostName to www.acme.com, HostAddress to ANY and PortNumber to 80.

Table 3 – 1 contains the parameters, grouped by logical function.

| Section | Parameter Name | Default Value | Description |
|---|---|---|---|
| NetInfo | HostName | none | hostname Web Listener is on, using the fully qualified host name, e.g.. www.acme.com<br>The Listener uses the primary Internet address if no hostname is specified. |
| | HostAddress | ANY | Internet address on which the Web Listener on a multi–homed host accepts connection. ANY indicates connections accepted on all interfaces (IP addresses). Leave this set to ANY unless you have a specific need to change it. |



MAC003095

ORCL00941771

| | | |
|---|---|---|
| PortNumber | 80 | TCP/ IP port number on which the Web Listener accepts requests. Note that port numbers less than 1024 require the Web Listener to run as root on many UNIX systems. 80 is the usual default for Web HTTP servers. |
| MaxConnectCount | 50 | Maximum number of simultaneous connections from users. The Web Listener ignores requests over this limit. Note there is a compile–time per–platform limit which may not be exceeded, no matter what value is specified here. |
| DNSResolution | NEVER | Controls DNS address to name resolution. If set to ALWAYS, the Web Listener always translates address to names. If set to LAZY, the Web Listener resolves names on demand only. If set to LAZY_WITH_CGI, the Web Listener also resolves on demand for CGI applications. If set to NEVER the Web Listener never resolves, even on demand (this provides the best performance). |
| SERVERPID | none | Name of file in which to store the Web Listener's process identifier (PID). |

Table 3 – 1 Oracle Web Listener Network Configuration Parameters

## Logging Configuration Parameters

Table 3 – 2 contains the Oracle Web Listener logging parameters.

| Section | Parameter Name | Default Value | Description |
|---|---|---|---|
| [Log] | LogDir | none | Name of directory to contain log files. This directory must exist before the Web Listener starts. |
| | LogInfoFile | none | Name of file in which to store routine log information. |
| | LogErrorFile | none | Name of file in which to log errors. |

Table 3 – 2 Oracle Web Listener Logging Configuration Parameters



MAC003096

ORCL00941772

Note: Be sure that the user ID under which the Web Listener runs has privileges to read and write to the log file directory.

## Directory Mapping Parameters

Directories are mapped individually in the [DirMaps] section. Each line looks like the following:

*Directory_Name* R|N|C *Virtual_Path_Name*

- The directory name on the left is the physical directory name on the Web Listener file system.
- The choice of R| N| C controls how the directory name is interpreted.
  - R means to map 'recursively', which means the directory is a tree of subdirectories, all of which are mapped into a tree starting at the specified virtual path name.
  - N means to map only the single directory specified, and no subdirectories.
  - C means that the directory contains CGI applications instead of files to be transmitted.
- Virtual_Path_Name is the location this directory should appear on the Web.

Consider the follow example of the directory mapping section of a Web Listener configuration file:

```
;
; directory mapping section of Web Listener configuration file
;
[DirMaps]
c:\html    R      /
```

For instance, in the above example, the directory c:\html and all subdirectories below it are mapped to the root directory (/) on the Web. So an access to the URL:

http://*listenername*/index.html

would actually read the following file on the Web Listener's disk.

c:\html\index.html

## File Cache Definitions

This section of the configuration file, starting with the key "[FileCache]" is simply a list of all the files that should be cached in memory. This



MAC003097

ORCL00941773

causes the Web Listener to keep the file open for the life of the Web Listener.

If a file is not listed here, it is not cached. When a non–cached file is requested by a client, the server maintains the resources to access that file only as long as the client refers to the file. The server will release the resources required to access a file requested by a client once there are no outstanding client references to the file.

There are three options for specifying files in the file cache.

- An explicit filename can be given.
- An incomplete filename ending in an asterisk can be used to indicate that all files that match the portion before the asterisk will be cached.
- A directory can be specified, in which case all its files and subdirectories will be cached.

The following example specifies that the specific file `index.html`, all files starting with `a` in the directory `/marketing`, and all files in the directory `/products` and its subdirectories should be cached.

```
;
; File Cache Definitions
;
[FileCache]
/index.html
/marketing/a*
/products
```

> Note: The filenames specified are virtual pathnames (the names that appear in URLs on the Web), not necessarily the physical names of the files on disk if the Directory Mapping feature has been used.

## Encoding Definitions

This section contains a list of file encodings that the Web Listener understands, and the file extensions that indicate a file is of the given type.

For example:

```
;
; encoding definitions
; List for each combination of encodings
; one or more unique extensions
;
[Encodings]
```



MAC003098

ORCL00941774

```
compress        Z
gzip            gz
```

## MIME Types Definitions

This section of the configuration file defines what Multimedia Internet Mail Extensions (MIME) file types the Web Listener recognizes, and how it maps the extensions of specific files to those types. This allows the proper information about file type to be transmitted to the client with the file data, which results in proper interpretation of the file on the client. The client may spawn a helper application to interpret the file if the type is not interpreted natively by the browser in use. This process is controlled entirely on the client side.

Note that files that do not map to any type are by default considered MIME type application/octet–stream which denotes a binary file.

Each line in this section consists of:

*MIME–type  list of file extensions mapping to that type*

For example:

```
[MIMEType]
text/html       htm html
image/jpeg      jpg jpeg
image/gif       gif
```

This example tells the Web Listener that files with extensions:

- htm and html are treated as MIME type text/html
- jpg and jpeg indicate an image/jpeg file
- gif indicates it is an image/gif file

These are all common MIME types that clients understand. There are many specialized types as well for conveying other forms of data, such as audio or video clips, and so forth.

## Language Extensions

This section defines filename extensions that the Web Listener recognizes and the languages and character sets to which those extensions are mapped. Extensions are case sensitive. Note that each mapping conveys both a language and a character set mapping. All language extensions that differ only in character set are equivalent for non–text files. The ability to specify a language and a character set is



MAC003099

ORCL00941775

important for languages which have more than one character set representation (for example, Arabic).

Each line consists of:

*language_ID character_set_ID list_of_file-extensions*

For example:

```
[LangExt]
en iso-8859-1   eng
en unicode-1-1  engUuc
fr-CA  iso-8859-1      frc
```

In this example, all files with the extension `eng` are mapped to English, ISO 8859–1 character set. Files with extension `engU` or `uc` are mapped to English, in the unicode character set. Files with extension `frc` are mapped to French Canadian language, ISO–8859–1 character sets.

The names of the character sets are as specified in RFC 1521, the language identifiers are defined in RFC 1766.

In order to use language mapping capability within the Web Listener, URLs must be specified without trailing extensions. The Web Listener adds the appropriate language/ character set extension as required if the client supports natural language negotiation.

For instance:

`http://www.oracle.com/index`

results in access to the file:

`/index.frc.html`

if the client had configured French Canadian as the preferred language.

A list of languages may be configured on the client, so that several languages may be specified and if any of them is present, the file will be sent according to the first match. Note that if a file has no language extension before its type extension, that file is assumed to be language neutral. The default character set is ISO–8859–1.

To summarize: if the files `/index.frc.html` and `/index.html` existed, users who specified French Canadian as a language preference would see the first version of the file, and all other users would see the second version.



MAC003100

ORCL00941776

## Basic and Digest Authentication Configuration

This section of the configuration file specifies the type of authentication accepted for a certain file access, and the parameters for that authentication type, including usernames, passwords, and file names.

There are several sections in the file, one for each type of security.

The concept in Basic and Digest authentication is the same. Users are listed in username/ password combinations, followed by specifying which users are in what groups. Finally, groups are assigned to security realms which are the parameters assigned to specific files to protect them.

Basic authentication is configured identically to Digest authentication, with the module name `Digest` above replaced with `Basic`.

Digest authentication is recommended wherever possible, to eliminate the possibility of passwords being intercepted on the network.

An example:

```
[Security]
        Digest {
                (Users)
                jjones: glorkz
                kmaco:  plugh
                ajames: foo

                (Groups)
                mktg:   ajames
                dev:    kmaco
                all:    jjones kmaco ajames

                (Realms)
                restricted:     all
                secret:         dev mktg
        }
```

This segment defines three users, (`jjones`, `kmaco`, `ajames`), with passwords (`glorkz`, `plugh`, `foo`) respectively. The group `all` has everyone in it, the groups `mktg` and `dev` have a single individual in each. The security realm `restricted` is accessible to all three people, but only the members of groups `dev` and `mktg` can access the `secret` realm.

In this situation, any file that was defined in the `restricted` realm would be accessible to the three users specified here (but no one else who used the same Web Listener), and any file in the `secret` realm would be accessible only to users `ajames` and `kmaco`.



MAC003101

ORCL00941777

## Network Address (IP) and Hostname Restriction Configuration

IP and hostname restriction is configured more simply. Hosts or IP addresses are assembled into groups, which can have specific files assigned to their access privileges.

To include a host or a group of hosts, specify the plus sign ( +) before the identifier for that host and exclude it by including a minus sign (–) before the identifier. In addition, the asterisk wildcard (*) can be used to match groups of hosts.

For example:

```
[Security]
        IP {
                st:     +144.25.16.*
                        +144.25.20.*
                dc:     +130.35.*.*
                        –130.35.1.1
        }
```

This would configure everything on the networks 144.25.16 and 144.25.20 to be in group st, and everything on network 130.35 except host 130.35.1.1 to be in group dc. If the client's address doesn't match any of these, access will be denied to any file specified to require st or dc security access.

The same technique is used with hostnames in the case of domain name restriction.

The following example configures all machines in the domain oracle.com to be in group oracle except machine www.oracle.com

```
[Security]
        Domain {
                oracle: +*.oracle.com
                        –www.oracle.com
        }
```

Once the security groups have been defined, you can assign specific protections to individual files.

## Assigning Security to Individual Files

When you assign protections, you specify the filename—possibly with wildcard characters to specify a group of files, or a directory name to include that directory and everything under it—followed by a list of security schemes and a parentheses–enclosed string. The string is a realm name for Digest or Basic authentication, and a group name for IP or domain restriction. The types may be combined, by requiring either scheme to be satisfied if the vertical bar ( | ) is used to separate security



MAC003102

ORCL00941778

schemes, or BOTH schemes must be satisfied if the ampersand (&) is used. If a combination of security schemes is used, one scheme must be an authentication type, and the other a restriction type.

For example:

```
[Protection]
/secret/                IP(st) | Basic(secret)
/company/*              Domain(acme)
```

This example protects everything under the directory /secret using either IP–based restriction from the st group or by anyone who can log into the secret realm using Basic authentication. All the files in /company are protected so that only people in the acme domain may see them.

Oracle Web Listener Miscellaneous Configuration Parameters

| Section Parameter Name | Default Value | Description |
|---|---|---|
| Server UserDir | none | Directory under a user's home directory searched when the URL /~username/ is received. This allows users to store their own home pages in their home directories (UNIX only). |
| InitialFile | initial | The file that the Listener looks for when a URL ending in a directory rather than a file is sent to the Listener. This is the 'index' file for the Web Listener. |
| UserDirInitialFile | initial.html | Default filename when the /~username/ construction is used in a URL with no file specified. |
| DefaultMIMEType | application/octet–stream | Default MIME type used by the Web Listener when the filename extension requested in a URL is not recognized. |
| DefaultCharset | iso–8859–1 | Default character set if none is inferred from the file extension. |
| PreferredLanguage | en | Language a Web Listener will prefer given a choice of languages when searching for a file. Default is English (en). |

| | | |
|---|---|---|
| ImageMap | none | Set this to 'map'. This is the extension the Web Listener expects imagemaps to have. |
| ServiceTimeout | none | On Windows NT only: Specifies the time in seconds that the NT control panel will wait for a pending start, stop, or continue command to complete before reporting a failure. |

Table 3 – 3 Oracle Web Listener Miscellaneous Listener Configuration Parameters



MAC003104

ORCL00941780

# Exhibit 5

# The Oracle WebServer: A Technical Discussion

Katrina Montinola, Oracle Corporation

Although the Internet has existed for over 30 years, it was generally regarded as the exclusive playground of the military, research and academic worlds until very recently, when the World Wide Web and NCSA Mosaic emerged. The World Wide Web is a set of techniques, protocols, technology and software that greatly simplify access to the vast amount of information available on the Internet. Together with NCSA Mosaic, the first graphical interface to the World Wide Web, it sparked the incredible growth of the Internet because it made navigating through the Internet as simple as pointing and clicking with a mouse.

## The Challenge of Managing Web-based Information

The exponential growth of the World Wide Web has not gone unnoticed in the corporate world. Corporations have started to realize its potential for information dissemination and electronic commerce and are scrambling to set up Web servers. Unfortunately, while putting up a pilot Web server is a relatively simple task, a number of challenges arise as these sites grow and start to be deployed in large-scale commercial projects. For instance, how does one store, manipulate, manage and retrieve large volumes of data? How does one provide secure access to information without compromise when there are 20 million potential users of this information? As corporations conduct electronic commerce over the Internet, how does one perform secure transactions and store results? As Web sites expand, how does one keep up with this growth and prevent their site from becoming an administrative nightmare?

## The Oracle WebServer

Oracle is the world's largest information management company. We have been solving similar problems for corporate data since the early 1980s, before the World Wide Web was even conceived. Since Web-based data has a lot in common with corporate data, we feel that the Oracle database can be just as effective at solving these problems. To address this need, we have developed the Oracle WebServer.

The Oracle WebServer is a World Wide Web server with a tightly-integrated Oracle7 database. It is targeted at users whose Web sites have grown too large and too complex to be managed effectively using traditional servers such as the NCSA and CERN servers. It consists of the following components: the Oracle Web Listener, the Oracle Web Agent, and the Oracle7 database. The Oracle WebServer provides a number of benefits beyond those of the traditional Web servers in many areas, the most important of which are:

Easy Access to Oracle Data - There is a large amount of data stored in Oracle databases that customers want to make available via the World Wide Web. Realizing this, we have provided a mechanism by which data stored in an Oracle7 database can be easily and efficiently transformed into HTML documents for distribution via the World Wide Web. This feature, called the Oracle Web Agent, will be available in Release 1.0 of the Oracle WebServer. High-Performance, Scalable, Portable, Commercial-quality Web Server - The Oracle WebServer will include the Oracle Web Listener, a full-fledged HTTP server engine that implements version 1.0 of the protocol as described in the RFC from the IETF HTTP Working Group. The server engine is designed to be a very reliable and high-performance solution suitable for high-traffic and mission-critical applications. Utilization of the Database for All Aspects of Web Server Management - In Release 2.0, which we have been developing in parallel with Release 1.0, a key differentiating factor between the Oracle WebServer and traditional Web servers will be the ability to store Web objects, such as HTML documents and GIF files, or pointers to Web objects, in the



ORCL01042968

database. Besides storing Web objects, we will also store the access log information and error log information in the database. Once the Web objects and other Web-related information are inside the database, Oracle's expertise in managing information can be put to work.

Although this paper will focus on providing technical information on the Oracle Web Listener and the Oracle Web Agent, two components of the Oracle WebServer Release 1.0, it will also touch briefly on Release 2.0 to give the reader a peek into the future of Web technology as we at Oracle see it.

## The Oracle Web Listener

Why use a commercial Web server when free ones are available from CERN and NCSA? This is a question that anyone planning to put up a Web site is sure to ask at some stage in the planning process. The most obvious reason: a commercial site that is running a mission-critical application needs a commercially-supported Web server because any bugs encountered in the server need to be fixed in a reasonable amount of time.

Of course, the decision to go with a commercial Web server begs the question: which commercial Web server is most suitable to the application? In most cases, this can only be answered by looking closely at the architecture of the Web servers under consideration. This is exactly what we did when we were shopping around for a third-party commercial Web server to include in the Oracle WebServer. The server we ultimately chose met our criteria for high-performance, scalability, portability and manageability.

For maximum performance and scalability, the Oracle Web Listener is designed to run as a single process with a single thread, as opposed to most servers which start a new thread or process whenever a new connection is made. This design improves performance in two ways: First, there is less overhead at the operating system level because the need for task switching and memory swapping is minimized. Second, the act of starting a new process or thread is time consuming and can significantly affect the latency between when a connection is established and when the server can begin to manipulate the data.

Oracle has always been known for the portability of its products, so this was a key feature for us. The Web server we ultimately chose was designed with portability in mind. Over 75% of the source code in the Oracle Web Listener is shared between the Windows NT and UNIX versions. The code was also designed to be easily portable to Windows95, should we decide to release a version on that platform in the future.

Since we are targeting large, mission-critical Web sites, manageability was also an important factor. The Oracle Web Listener will include Simple Network Management Protocol (SNMP) support. In the first release, this will be limited to basic "I'm alive" messages. However, we will implement the different portions of the Management Image Base (MIB) for HTTP servers as it gets standardized.

## The Oracle Web Agent

With approximately 35% of the relational database market, there is a huge amount of data stored in Oracle databases today. A significant percentage of this data would be even more useful and valuable if it could be accessed via the World Wide Web. To assist our customers with this task, we have developed the Oracle Web Agent. The Web Agent is a component of the Oracle WebServer that facilitates the creation of dynamic HTML documents via a language already familiar to most Oracle users: PL/SQL.

In addition to existing corporate data, the Oracle database could also be used to store data meant



ORCL01042969

exclusively for the Web. If this data resides in tables in a relational database and simply gets transformed into dynamic HTML documents when accessed, it will be much simpler to manage and manipulate. Furthermore, dynamic documents provide a greater degree of customization, making it possible to tailor a document to the specific user requesting it. The Oracle Web Agent can be used to create these dynamic HTML documents quickly and easily in PL/SQL.

The Oracle Web Agent consists of the following:

A "C" program named owa which serves as the actual CGI program that the Web Listener executes when a POST or GET request is received for a dynamic document. This program makes OCI calls to log into the database and execute the user's PL/SQL stored procedures.

A PL/SQL package called wow which contains utility functions to set up the CGI environment variables, obtain the value of a specific environment variable, and extract the output generated by the user's PL/SQL code from a PL/SQL table.

An optional, but strongly recommended set of PL/SQL packages known as the WebServer Developer's Toolkit that ease the generation of HTML tags from within the user's PL/SQL code. HTP and HTF, which are two of the packages in this toolkit, automatically generate the HTML tags and put the document into the PL/SQL table where it gets retrieved by the OCI program called owa.

A set of HTML forms and corresponding code to ease the task of configuring the CGI applications written in PL/SQL with the Web Agent.

A PL/SQL development environment which allows the programmer to compile and submit code into the database directly from within an HTML form. This is a Release 2.0 feature.

The Oracle Web Agent provides an easy-to-use environment for building CGI applications in PL/SQL, which are then stored in the Oracle7 database. A programmer who has chosen PL/SQL as the language in which to implement a CGI application need only worry about implementing the logic specific to the application itself, because the Oracle Web Agent provides utilities which take care of the repetitive tasks associated with deploying a CGI application.

Why chose PL/SQL in the first place? Two reasons: First, PL/SQL is compiled and stored inside the database, thus PL/SQL code tends to execute faster than regular SQL code that needs to be parsed and interpreted before executing. Second, since PL/SQL code resides inside the database, it is completely portable to any platform that Oracle7 runs on. Porting PL/SQL code is simply a matter of moving the PL/SQL packages from one database to another in the same way that moving regular database data is accomplished. Step 2 Step 3 Step 5 Step 4 Step 1 Step 6

Web client

The figure above illustrates the sequence of events that take place in the Oracle WebServer when a typical GET request is received for a dynamic HTML document produced via the Oracle Web Agent. .The Web client issues a GET request for an HTML document which is actually a dynamic document produced by PL/SQL code residing in the database. .The Web Listener spawns a "C" stub program called owa. .owa makes OCI calls that log into the database and execute the PL/SQL procedure which produces the HTML document. It knows which PL/SQL procedure to execute because the name of the PL/SQL procedure is embedded in the URL. .owa takes the HTML document produced by the PL/SQL procedure from a PL/SQL table where the document has been placed by the HTP/HTF calls made by the

PL/SQL procedure. .owa passes it back to the Web Listener via standard output. .The Web Listener passes it back to the client via the HTTP protocol.

To get a better technical understanding of the Oracle Web Agent, let's examine a number of it's key features in greater detail.

I. CGI Stub

Since PL/SQL procedures are stored inside the database, there is no way for a Web server to directly execute them when the request from the browser is received. Furthermore, there is no way for the PL/SQL code to send the HTML document generated to standard output, which is where the Web server is expecting it to be. Thus, there is a need for a program which handles these tasks. The Oracle Web Agent includes a "C" program called owa which does the following using OCI calls : .Extracts the database service name embedded in the URL and logs on to the database using the userid/password defined by the service. Database services are defined in a file called owa.ora. .Sets up the parameters passed from the Web server so that they are accessible from within the database through a PL/SQL function call similar to getenv() in UNIX. .Executes the user's PL/SQL function which extracts the relevant data from the database and formats it into an HTML document that is stored in a PL/SQL table. .Checks the return code of the user's PL/SQL function. If it is zero, it takes the formatted HTML document from the PL/SQL table and passes it back to the web server via standard output. If it is non-zero, it returns the standard error HTML document supplied by the user or a default if the user doesn't supply one.

Thus, it is not necessary for the PL/SQL programmer to be intimately familiar with how CGI works. owa takes care of all those details and allows the programmer to concentrate on developing the logic to extract the proper data from the database. The user's PL/SQL code need only make function calls to extract the values of the necessary CGI environment variables, query the database, and format the results returned from the database into an HTML document. Everything else is taken care of by owa.

Both the database service name and the user's PL/SQL function name are embedded in the URL, which is of the following format:

http://myhostname:port/.../myservicename/owa/myplsqlname?parameterlist

URL section Description

| | |
|---|---|
| http | denotes the protocol used to obtain the document from the WebServer |
| myhost | denotes the hostname or IP address of the machine the WebServer resides in |
| port | an optional parameter, this specifies the TCP port number that the Web Listener is listening on. If none is specified, it defaults to 80 |
| myservicename | denotes the database service name to use when logging on to Oracle7. The database service name immediately precedes 'owa' but doesn't have to directly follow the hostname/port section of the URL. |
| owa | must always exist. This instructs the Web Listener to spawn the owa executable |
| myplsqlname | denotes the name of the PL/SQL procedure that gets spawned when owa logs on to Oracle7 |
| parameterlist | If a '?' exists, any text following it will be passed by owa to the PL/SQL procedure it just invoked. This will only exist if the GET method is used. |

II. HTML tags

HTML documents are generated by inserting the necessary HTML tags around the text output which Web browsers can interpret. The HTP/HTF PL/SQL packages which are part of the Oracle WebServer Developer's Toolkit greatly simplify this task. The PL/SQL programmer need only call these functions at the appropriate place in the code with the actual text as a parameter. Then, the HTP (HyperText Procedure) and corresponding HTF (HyperText Function) take care of generating the correct tags around the text and putting it into a PL/SQL table.

It is important to note that the HTP/HTF packages do not eliminate the need for the PL/SQL programmer to know HTML syntax. The programmer must still realize that an anchor tag is needed to create a hyperlink, for example. What the HTP/HTF packages do is automate the process of creating these tags and putting the document in the PL/SQL table.

III. Development Environment

PL/SQL code is typically developed using SQL*Plus or an embedded development environment in tools such as Oracle Forms 4.5. Since CGI application programmers may not have access to these tools or may not be familiar with them, the Oracle Web Agent will provide a development environment accessible via a Web browser. The Oracle Web Agent will include an HTML form which will allow programmers to type in PL/SQL code. The form will have a push button to compile the code and submit it into the database. If there are errors in the compilation phase, these will be returned to the browser as an HTML document. This is a Release 2.0 feature.

IV. Error Handling

A CGI application written in PL/SQL can generate two broad types of errors: application errors and system errors. Application errors are specific to the application, and are generally meaningful to the end user. For example, if the query returns no rows, the PL/SQL programmer could add code to generate an HTML document that said, "The item you ordered is temporarily out of stock. Please try again later." These errors are transparent to owa because owa does not read HTML documents to determine their content. As far as owa is concerned, if the user's PL/SQL code generates an HTML document, the operation was successful. owa determines whether an HTML document has been generated by checking the return code from the original call to execute the user's PL/SQL program. If the return code is zero, an HTML document has been generated in the PL/SQL table.

System errors are errors encountered by owa itself. These are errors that prevent owa from executing the user's PL/SQL code, or failure of the user's PL/SQL code to generate an HTML document. A user's PL/SQL code signals it's failure to generate an HTML document by returning an error to owa.

An example of the first type of system error would be the failure of owa to log on to the database. An example of the second type of system error would be an error in the SQL query because the table does not exist. In the latter case, the programmer could write code to format an HTML document with the specific reason for the error, but it would be more convenient for the programmer if there were a standard HTML document that could be returned for errors such as these. The Oracle Web Agent provides a mechanism for doing just that. To avail himself or herself of this mechanism, the programmer returns an error to owa (see Section I, step 4). owa will either return a default error message, or return a predefined error message when it gets the non-zero return code.

The Oracle WebServer: The Future



ORCL01042972

Although the Oracle Web Agent greatly simplifies the development of CGI applications that need access to data stored in an Oracle7 database, we feel that Web servers can benefit from even tighter integration with the database. By this we mean the following:

Direct access to PL/SQL stored procedures

Although CGI works pretty well for small Web servers with relatively light loads, it doesn't provide adequate performance and doesn't scale well for Web servers that are heavily accessed. This is because CGI requires the Web server to spawn a separate process for each document requested. When Web servers start experiencing hundreds of simultaneous requests, the number of spawned processes can easily bog down the machine. Realizing this, in Release 2.0 of the Oracle WebServer, we will remove the Oracle Web Agent's reliance on CGI. owa will no longer be a separate "C" executable that gets spawned by the Web server. It will be linked directly into the Web server and will pass on the name of the PL/SQL procedure to be executed to a configurable number of processes that are always connected to the Oracle7 database. Besides providing a much more scaleable solution, this method will also enjoy increased performance, because spawning a process and logging on to the Oracle7 database are some of the slowest operations in a typical transaction.

It is important to note, however, that we are not removing the CGI interface altogether. We realize that it is important to continue to support CGI because it is an open standard that will allow our Web server to support any application written for any other Web server that supports this standard. We are merely eliminating the need to go through CGI for applications written with the Oracle Web Agent to provide increased performance and scalability.

Replacing the File System Itself with the Database

Anything that a traditional Web server would use the file system for, the Oracle WebServer Release 2.0 will use the database. This will include storage of static HTML documents and other Web objects, storage of access log information, and storage of error log information. The benefits of storing logging information in the database are immediately apparent: it will be possible to formulate much more sophisticated queries using SQL to generate reports on the actual usage and hit rate of the Web server. However, the benefits of storing actual Web objects in the database are not as obvious. Thus, I'd like to focus on this area for the rest of this section.

Although the Oracle Web Listener, which is a component of the Oracle WebServer, is a full-fledged, industrial-strength HTTP server that can serve HTML documents off the file system like any other Web server, most of the benefits of the Oracle WebServer will not be realized until the database is utilized to store Web objects. Storing Web objects in the database has the following advantages:

Since a list of objects or the objects themselves reside in the database, it will be possible to use the database to enforce additional security using the concept of database roles. The web server administrator will be able to restrict groups of objects to certain users and have these users administer their own groups. The designated administrator for a particular group will be able to add users and objects to the group, but will not be able to tamper with objects in a group with a higher privilege. This allows a finer granularity of security than is available on traditional web servers.

It will be possible to write SQL and PL/SQL code to retrieve information about the objects in the Web server. This opens up a whole range of possibilities such as the ability to maintain different versions of objects much more easily, the ability to locate specific objects more easily, etc.



ORCL01042973

Searching and retrieving Web objects based on their content will be possible. This is slow and quite cumbersome to do with flat files.

Realizing that for large Web sites it will be a big task to load all existing documents into the database, we are providing the option to leave the documents in the file system and simply create pointers to them in the database. Creating pointers in the database will still bring you the first two advantages described above. Furthermore, for extremely large documents, storing them in the file system with pointers in the database may be preferable from a performance standpoint. Fetching a 10 megabyte document from the database, for example, could be significantly slower than fetching it from the file system. For smaller documents, this performance hit is small enough that the ability to do sophisticated searches based on the content of the document and increased security would outweigh the performance degradation.

Integration with Oracle ConText

The World Wide Web's greatest asset is the wealth of information available to anyone at the click of a mouse button. However, the Web may be the classic case of "too much of a good thing" because it is becoming increasingly difficult to find the information needed due to the sheer volume of information available. Oracle ConText, which will be integrated into the Oracle WebServer, is the language-processing core of Oracle's solution to this information overload.

ConText is a natural language technology that uses the linguistic information contained in every word, sentence, and paragraph to identify the themes and content of any text. With this knowledge, ConText can facilitate automatic classification, visual browsing, and easy management of ever-increasing volumes of electronic information. Its analysis delivers insights into documents that statistical methods cannot:

Main themes representing a selection of text form a "document fingerprint" for classifying or retrieving.

Prominent ideas can be highlighted while details are hidden, for summaries or abstracts.

Areas of text discussing similar themes can be identified, collapsed, or indexed, for hyperlink navigation or finding information quickly.

ConText is unique in that it appears to understand the English language: it can distinguish between a document that merely contains the term "Computer Science" 100 times in the document, and a document that has Computer Science as its main theme, for example. The actual words, "Computer Science", may not even appear anywhere in the latter document.

This capability is extremely useful for large Web sites that want to be able to automatically classify their documents to assist users in finding what they need quickly and easily. After all, information is only valuable if one can find it when it is needed.

Conclusion

We at Oracle believe that in the future, it will be difficult to envision a commercial World Wide Web site without a backend database to manage Web content as well as other information about the Web site, such as access statistics. The Oracle WebServer is only the first product from Oracle aimed at addressing this need. Stay tuned for more!



ORCL01042974

# Exhibit 6

**THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY**

# Exhibit 7

**THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY**

# Exhibit 8

# Oracle® Database

Application Developer's Guide - Fundamentals

10g Release 2 (10.2)

**B14251-01**

November 2005

ORACLE®

Oracle Database Application Developer's Guide - Fundamentals, 10g Release 2 (10.2)

B14251-01

Copyright © 1996, 2005, Oracle. All rights reserved.

Primary Author: Lance Ashdown

Contributing Authors: D. Adams, M. Cowan, R. Moran, J. Melnick, E. Paapanen, J. Russell, R. Strohm

Contributors: D. Alpern, G. Arora, C. Barclay, D. Bronnikov, T. Chang, M. Davidson, G. Doherty, D. Elson, A. Ganesh, M. Hartstein, J. Huang, N. Jain, R. Jenkins Jr., S. Kotsovolos, S. Kumar, C. Lei, D. Lorentz, V. Moore, J. Muller, R. Murthy, R. Pang, B. Sinha, S. Vemuri, W. Wang, D. Wong, A. Yalamanchi, Q. Yu

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software—Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

# 1

# Orientation to Oracle Programmatic Environments

This chapter contains these topics:


- Overview of Oracle Application Development
- Overview of PL/SQL
- Overview of Java Support Built Into the Database
- Overview of Pro*C/C++
- Overview of Pro*COBOL
- Overview of OCI and OCCI
- Overview of Oracle Data Provider for .NET (ODP.NET)
- Overview of Oracle Objects for OLE (OO4O)
- Choosing a Programming Environment


## Overview of Oracle Application Development

As an application developer, you have many choices when writing a program to interact with an Oracle database.

### Client/Server Model

In a traditional client/server program, the code of your application runs on a machine other than the database server. Database calls are transmitted from this client machine to the database server. Data is transmitted from the client to the server for insert and update operations and returned from the server to the client for query operations. The data is processed on the client machine. Client/server programs are typically written by using precompilers, whereas SQL statements are embedded within the code of another language such as C, C++, or COBOL.

### Server-Side Coding

You can develop application logic that resides entirely inside the database by using triggers that are executed automatically when changes occur in the database or stored procedures that are called explicitly. Off-loading the work from your application lets you reuse code that performs verification and cleanup and control database operations from a variety of clients. For example, by making stored procedures callable through a Web server, you can construct a Web-based user interface that performs the same functions as a client/server application.

### Two-Tier Versus Three-Tier Models

Client/server computing is often referred to as a **two-tier model**: your application communicates directly with the database server. In the **three-tier model**, a separate **application server** processes the requests. The application server might be a basic Web server, or might perform advanced functions like caching and load-balancing. Increasing the processing power of this middle tier lets you lessen the resources needed by client systems, resulting in a **thin client** configuration in which the client machine might need only a Web browser or other means of sending requests over the TCP/IP or HTTP protocols.

### User Interface

The interface that your application displays to end users depends on the technology behind the application as well as the needs of the users themselves. Experienced users can enter SQL commands that are passed on to the database. Novice users can be shown a graphical user interface that uses the graphics libraries of the client system (such as Windows or X-Windows). Any of these traditional user interfaces can also be provided in a Web browser by means of HTML and Java.

### Stateful Versus Stateless User Interfaces

In traditional client/server applications, the application can keep a record of user actions and use this information over the course of one or more sessions. For example, past choices can be presented in a menu so that they do not have to be entered again. When the application is able to save information in this way, the application is considered **stateful**.

Web or thin-client applications that are **stateless** are easier to develop. Stateless applications gather all the required information, process it using the database, and then start over with the next user. This is a popular way to process single-screen requests such as customer registration.

There are many ways to add stateful behavior to Web applications that are stateless by default. For example, an entry form on one Web page can pass information to subsequent Web pages, allowing you to construct a wizard-like interface that remembers the user's choices through several different steps. Cookies can be used to store small items of information on the client machine, and retrieve them when the user returns to a Web site. Servlets can be used to keep a database session open and store variables between requests from the same client.

## Overview of PL/SQL

This section contains the following topics:

- What Is PL/SQL?
- Advantages of PL/SQL
- PL/SQL Web Development Tools

## What Is PL/SQL?

PL/SQL is Oracle's procedural extension to SQL, the standard database access language. It is an advanced 4GL (fourth-generation programming language), which means that it is an application-specific language. PL/SQL and SQL have built-in treatment of the relational database domain.

In PL/SQL, you can manipulate data with SQL statements and control program flow with procedural constructs such as loops. You can also do the following:

- Declare constants and variables
- Define procedures and functions
- Use collections and object types
- Trap runtime errors

Applications written in any of the Oracle programmatic interfaces can call PL/SQL stored procedures and send blocks of PL/SQL code to Oracle Database for execution. 3GL applications can access PL/SQL scalar and composite datatypes through host variables and implicit datatype conversion. A 3GL language is easier than assembler language for a human to understand and includes features such as named variables. Unlike 4GL, it is not specific to an application domain.

Example 1–1 provides an example of a simple PL/SQL procedure. The procedure `debit_account` withdraws money from a bank account. It accepts an account number and an amount of money as parameters. It uses the account number to retrieve the account balance from the database, then computes the new balance. If this new balance is less than zero, then the procedure jumps to an error routine; otherwise, it updates the bank account.

***Example 1–1 Simple PL/SQL Example***

```
PROCEDURE debit_account (p_acct_id INTEGER, p_debit_amount REAL)
IS
   v_old_balance  REAL;
   v_new_balance  REAL;
   e_overdrawn    EXCEPTION;
BEGIN
   SELECT bal
     INTO v_old_balance
   FROM accts
   WHERE acct_no = p_acct_id;
   v_new_balance := v_old_balance - p_debit_amount;
   IF v_new_balance < 0 THEN
      RAISE e_overdrawn;
   ELSE
      UPDATE accts SET bal = v_new_balance
         WHERE acct_no = p_acct_id;
   END IF;
   COMMIT;
EXCEPTION
   WHEN e_overdrawn THEN
      -- handle the error
END debit_account;
```

**See Also:**

- *Oracle Database PL/SQL User's Guide and Reference*
- *Oracle Database SQL Reference*

## Advantages of PL/SQL

PL/SQL is a portable, high-performance transaction processing language with the following advantages:

- Integration with Oracle Database
- High Performance
- High Productivity

- Scalability
- Manageability
- Object-Oriented Programming Support
- Portability
- Security
- Built-In Packages

### Integration with Oracle Database

PL/SQL enables you use all of the Oracle Database SQL data manipulation, cursor control, and transaction control statements. PL/SQL also supports the SQL functions, operators, and pseudocolumns. So, you can manipulate data in Oracle Database flexibly and safely.

PL/SQL supports all SQL datatypes. Combined with the direct access that SQL provides, these shared datatypes integrate PL/SQL with the Oracle Database data dictionary.

PL/SQL supports Dynamic SQL, which is a programming technique that enables you to build and process SQL statements "on the fly" at run time. It gives PL/SQL flexibility comparable to scripting languages such as Perl, Korn shell, and Tcl.

The `%TYPE` and `%ROWTYPE` attributes enable your code to adapt as table definitions change. For example, the `%TYPE` attribute declares a variable based on the type of a database column. If the column datatype changes, then the variable uses the correct type at runtime. This provides data independence and reduces maintenance costs.

### High Performance

If your application is database intensive, then you can use PL/SQL blocks to group SQL statements before sending them to Oracle Database for execution. This coding strategy can drastically reduce the communication overhead between your application and Oracle Database.

PL/SQL stored procedures are compiled once and stored in executable form, so procedure calls are quick and efficient. A single call can start a compute-intensive stored procedure, reducing network traffic and improving round-trip response times. Executable code is automatically cached and shared among users, lowering memory requirements and invocation overhead.

### High Productivity

PL/SQL adds procedural capabilities such as Oracle Forms and Oracle Reports. For example, you can use an entire PL/SQL block in an Oracle Forms trigger instead of multiple trigger steps, macros, or user exits.

PL/SQL is the same in all environments. As soon as you master PL/SQL with one Oracle tool, you can transfer your knowledge to others, and so multiply the productivity gains. For example, scripts written with one tool can be used by other tools.

### Scalability

PL/SQL stored procedures increase scalability by centralizing application processing on the server. Automatic dependency tracking helps you develop scalable applications.

The shared memory facilities of the shared server enable Oracle Database to support many thousands of concurrent users on a single node. For more scalability, you can use the Oracle Connection Manager to multiplex network connections.

### Manageability

After being validated, you can use a PL/SQL stored procedure in any number of applications. If its definition changes, then only the procedure is affected, not the applications that call it. This simplifies maintenance and enhancement. Also, maintaining a procedure on the Oracle Database is easier than maintaining copies on various client machines.

### Object-Oriented Programming Support

You can use PL/SQL object types and collections for object-oriented programming.

**Object Types** An object type is a user-defined composite datatype that encapsulates a data structure along with the functions and procedures needed to manipulate the data. The variables that form the data structure are called attributes. The functions and procedures that characterize the behavior of the object type are called methods, which you can implement in PL/SQL.

Object types are an ideal object-oriented modeling tool, which you can use to reduce the cost and time required to build complex applications. Besides allowing you to create software components that are modular, maintainable, and reusable, object types allow different teams of programmers to develop software components concurrently.

**Collections** A collection is an ordered group of elements, all of the same type (for example, the grades for a class of students). Each element has a unique subscript that determines its position in the collection. PL/SQL offers two kinds of collections: nested tables and varrays (variable-size arrays).

Collections work like the set, queue, stack, and hash table data structures found in most third-generation programming languages. Collections can store instances of an object type and can also be attributes of an object type. Collections can be passed as parameters. So, you can use them to move columns of data into and out of database tables or between client-side applications and stored subprograms. You can define collection types in a PL/SQL package, then use the same types across many applications.

### Portability

Applications written in PL/SQL can run on any operating system and hardware platform on which Oracle Database runs. You can write portable program libraries and reuse them in different environments.

### Security

PL/SQL stored procedures enable you to divide application logic between the client and the server, which prevents client applications from manipulating sensitive Oracle Database data. Database triggers written in PL/SQL can prevent applications from making specified updates and can audit user queries.

You can restrict access to Oracle Database data by allowing users to manipulate it only through stored procedures that have a restricted set of privileges. For example, you can grant users access to a procedure that updates a table but not grant them access to the table itself.

**See Also:** *Oracle Database Security Guide* for details on database security features

### Built-In Packages

A package is an encapsulated collection of related program objects stored together in the database. Program objects are procedures, functions, variables, constants, cursors, and exceptions.

The following packages are especially useful in application development for Oracle Database:

- `DBMS_PIPE` is used to communicate between sessions.
- `DBMS_ALERT` is used to broadcast alerts to users.
- `DBMS_LOCK` and `DBMS_TRANSACTION` are used for lock and transaction management.
- `DBMS_AQ` is used for Advanced Queuing.
- `DBMS_LOB` is used to manipulate large objects.
- `DBMS_ROWID` is used for employing `ROWID` values.
- `UTL_RAW` is the RAW facility.
- `UTL_REF` is for work with `REF` values.

The following packages are useful for server manageability:

- `DBMS_SESSION` is for session management by DBAs.
- `DBMS_SPACE` and `DBMS_SHARED_POOL` provide space information and reserve shared pool resources.
- `DBMS_JOB` is used to schedule jobs in the server.

## PL/SQL Web Development Tools

Oracle Database provides built-in tools and technologies that enable you to deploy PL/SQL applications over the Web. Thus, PL/SQL serves as an alternative to Web application frameworks such as CGI.

The **PL/SQL Web Toolkit** is a set of PL/SQL packages that you can use to develop stored procedures that can be invoked by a Web client. The PL/SQL Gateway enables an HTTP client to invoke a PL/SQL stored procedure through mod_plsql, which is a plug-in to Oracle HTTP Server. This module performs the following actions:

1. Translates a URL passed by a browser client
2. Calls an Oracle Database stored procedure with the parameters in the URL
3. Returns output (typically HTML) to the client

**See Also:** Chapter 11, "Developing Applications with the PL/SQL Web Toolkit" to learn how to use PL/SQL in Web development

# Overview of Java Support Built Into the Database

This section provides an overview of built-in database features that support Java applications. The database includes the core JDK libraries such as `java.lang`, `java.io`, and so on. The database supports client-side Java standards such as JDBC

and SQLJ, and provides server-side JDBC and SQLJ drivers that allow data-intensive Java code to run within the database.

This section contains the following topics:


- Overview of Oracle JVM
- Overview of Oracle Extensions to JDBC
- Overview of Oracle SQLJ
- Overview of Oracle JPublisher
- Overview of Java Stored Procedures
- Overview of Oracle Database Web Services
- Overview of Writing Procedures and Functions in Java


**See Also:**

- *Oracle Database Concepts* for background information about Java and how the database supports it
- *Oracle Database Java Developer's Guide*
- *Oracle Database JDBC Developer's Guide and Reference*
- *Oracle Database JPublisher User's Guide*

## Overview of Oracle JVM

Oracle JVM, the Java Virtual Machine provided with the Oracle Database, is compliant with the J2SE version 1.4.x specification and supports the database session architecture.

Any database session can activate a dedicated JVM. All sessions share the same JVM code and statics; however, private states for any given session are held, and subsequently garbage collected, in an individual session space.

This design provides the following benefits:

- Java applications have the same session isolation and data integrity as SQL operations.
- There is no need to run Java in a separate process for data integrity.
- Oracle JVM is a robust JVM with a small memory footprint.
- The JVM has the same linear Symmetric Multiprocessing (SMP) scalability as the database and can support thousands of concurrent Java sessions.

Oracle JVM works consistently with every platform supported by Oracle Database. Java applications that you develop with Oracle JVM can easily be ported to any supported platform.

Oracle JVM includes a deployment-time native compiler that enables Java code to be compiled once, stored in executable form, shared among users, and invoked more quickly and efficiently.

Security features of the database are also available with Oracle JVM. Java classes must be loaded in a database schema (by using Oracle JDeveloper, a third-party IDE, SQL*Plus, or the loadjava utility) before they can be invoked. Java class invocation is secured and controlled through database authentication and authorization, Java 2 security, and invoker's or definer's rights.

## Overview of Oracle Extensions to JDBC

JDBC (Java Database Connectivity) is an API (Applications Programming Interface) that allows Java to send SQL statements to an object-relational database such as Oracle Database.

The JDBC standard defines four types of JDBC drivers:

- Type 1. A JDBC-ODBC bridge. Software must be installed on client systems.
- Type 2. Native methods (calls C or C++) and Java methods. Software must be installed on the client.
- Type 3. Pure Java. The client uses sockets to call middleware on the server.
- Type 4. The most pure Java solution. Talks directly to the database by using Java sockets.

JDBC is based on the X/Open SQL Call Level Interface, and complies with the SQL92 Entry Level standard.

You can use JDBC to do dynamic SQL. In dynamic SQL, the embedded SQL statement to be executed is not known before the application is run and requires input to build the statement.

The drivers that are implemented by Oracle have extensions to the capabilities in the JDBC standard that was defined by Sun Microsystems. Oracle's implementations of JDBC drivers are described in the following sections. Oracle Database support of and extensions to various levels of the JDBC standard are described in "Oracle Database Extensions to JDBC Standards" on page 1-9.

### JDBC Thin Driver

The JDBC thin driver is a Type 4 (100% pure Java) driver that uses Java sockets to connect directly to a database server. It has its own implementation of a Two-Task Common (TTC), a lightweight implementation of TCP/IP from Oracle Net. It is written entirely in Java and is therefore platform-independent.

The thin driver does not require Oracle software on the client side. It does need a TCP/IP listener on the server side. Use this driver in Java applets that are downloaded into a Web browser or in applications for which you do not want to install Oracle client software. The thin driver is self-contained, but it opens a Java socket, and thus can only run in a browser that supports sockets.

### JDBC OCI Driver

The OCI driver is a Type 2 JDBC driver. It makes calls to the OCI (Oracle Call Interface) written in C to interact with Oracle Database, thus using native and Java methods.

The OCI driver allows access to more features than the thin driver, such as Transparent Application Fail-Over, advanced security, and advanced LOB manipulation.

The OCI driver provides the highest compatibility between different Oracle Database versions. It also supports all installed Oracle Net adapters, including IPC, named pipes, TCP/IP, and IPX/SPX.

Because it uses native methods (a combination of Java and C) the OCI driver is platform-specific. It requires a client installation of version Oracle8*i* or later including Oracle Net, OCI libraries, CORE libraries, and all other dependent files. The OCI driver usually executes faster than the thin driver.

The OCI driver is not appropriate for Java applets, because it uses a C library that is platform-specific and cannot be downloaded into a Web browser. It is usable in J2EE components running in middle-tier application servers, such as Oracle Application Server. Oracle Application Server provides middleware services and tools that support access between applications and browsers.

### JDBC Server-Side Internal Driver

The JDBC server-side internal driver is a Type 2 driver that runs inside the database server, reducing the number of round-trips needed to access large amounts of data. The driver, the Java server VM, the database, the Java native compiler which speeds execution by as much as 10 times, and the SQL engine all run within the same address space.

This driver provides server-side support for any Java program used in the database: SQLJ stored procedures, functions, triggers, and Java stored procedures. You can also call PL/SQL stored procedures, functions, and triggers.

The server driver fully supports the same features and extensions as the client-side drivers.

### Oracle Database Extensions to JDBC Standards

Oracle Database includes the following extensions to the JDBC 1.22 standard:

- Support for Oracle datatypes
- Performance enhancement by row prefetching
- Performance enhancement by execution batching
- Specification of query column types to save round-trips
- Control of `DatabaseMetaData` calls

Oracle Database supports all APIs from the JDBC 2.0 standard, including the core APIs, optional packages, and numerous extensions. Some of the highlights include datasources, JTA, and distributed transactions.

Oracle Database supports the following features from the JDBC 3.0 standard:

- Support for JDK 1.4.
- Toggling between local and global transactions.
- Transaction savepoints.
- Reuse of prepared statements by connection pools.

### Sample JDBC 2.0 Program

The following example shows the recommended technique for looking up a data source using JNDI in JDBC 2.0:

```
// import the JDBC packages
import java.sql.*;
import javax.sql.*;
import oracle.jdbc.pool.*;
...
   InitialContext ictx = new InitialContext();
   DataSource ds = (DataSource)ictx.lookup("jdbc/OracleDS");
   Connection conn = ds.getConnection();
   Statement stmt = conn.createStatement();
   ResultSet rs = stmt.executeQuery("SELECT last_name FROM employees");
```

```
   while ( rs.next() ) {
   out.println( rs.getString("ename") + "<br>");
   }
conn.close();
```

### Sample Pre-2.0 JDBC Program

The following source code registers an Oracle JDBC thin driver, connects to the database, creates a `Statement` object, executes a query, and processes the result set.

The `SELECT` statement retrieves and lists the contents of the `last_name` column of the `hr.employees` table.

```
import java.sql.*
import java.math.*
import java.io.*
import java.awt.*

class JdbcTest {
  public static void main (String args []) throws SQLException {
    // Load Oracle driver
    DriverManager.registerDriver (new oracle.jdbc.OracleDriver());

    // Connect to the local database
    Connection conn =
      DriverManager.getConnection ("jdbc:oracle:thin:@myhost:1521:orcl",
                                   "hr", "hr");

    // Query the employee names
    Statement stmt = conn.createStatement ();
    ResultSet rset = stmt.executeQuery ("SELECT last_name FROM employees");

    // Print the name out
    while (rset.next ())
      System.out.println (rset.getString (1));
    // Close the result set, statement, and the connection
    rset.close();
    stmt.close();
    conn.close();
  }
}
```

One Oracle Database extension to the JDBC drivers is a form of the `getConnection()` method that uses a `Properties` object. The `Properties` object lets you specify user, password, and database information as well as row prefetching and execution batching.

To use the OCI driver in this code, replace the `Connection` statement with the following, where `MyHostString` is an entry in the `tnsnames.ora` file:

```
Connection conn = DriverManager.getConnection ("jdbc:oracle:oci8:@MyHostString",
    "hr", "hr");
```

If you are creating an applet, then the `getConnection()` and `registerDriver()` strings will be different.

### JDBC in SQLJ Applications

JDBC code and SQLJ code (see "Overview of Oracle SQLJ" on page 1-11) interoperate, allowing dynamic SQL statements in JDBC to be used with both static and dynamic SQL statements in SQLJ. A SQLJ iterator class corresponds to the JDBC result set.

**See Also:** *Oracle Database JDBC Developer's Guide and Reference* for more information on JDBC

## Overview of Oracle SQLJ

SQLJ is an ANSI SQL-1999 standard for embedding SQL statements in Java source code. SQLJ provides a simpler alternative to JDBC for both client-side and server-side SQL data access from Java.

A SQLJ source file contains Java source with embedded SQL statements. Oracle SQLJ supports dynamic as well as static SQL. Support for dynamic SQL is an Oracle extension to the SQLJ standard.

---

**Note:** The term "SQLJ," when used in this manual, refers to the Oracle SQLJ implementation, including Oracle SQLJ extensions.

---

Oracle Database provides a translator and a run time driver to support SQLJ. The SQLJ translator is 100% pure Java and is portable to any JVM that is compliant with JDK version 1.1 or higher.

The Oracle SQLJ translator performs the following tasks:

- Translates SQLJ source to Java code with calls to the SQLJ run time driver. The SQLJ translator converts the source code to pure Java source code and can check the syntax and semantics of static SQL statements against a database schema and verify the type compatibility of host variables with SQL types.
- Compiles the generated Java code with the Java compiler.
- (Optional) Creates profiles for the target database. SQLJ generates "profile" files with customization specific to Oracle Database.

Oracle Database supports SQLJ stored procedures, functions, and triggers which execute in the Oracle JVM. SQLJ is integrated with JDeveloper. Source-level debugging support for SQLJ is available in JDeveloper.

The following is an example of a simple SQLJ executable statement, which returns one value because `employee_id` is unique in the `employee` table:

```
String name;
#sql  { SELECT first_name INTO :name FROM employees WHERE employee_id=112 };
System.out.println("Name is " + name + ", employee number = " + employee_id);
```

Each host variable (or qualified name or complex Java host expression) included in a SQL expression is preceded by a colon (`:`). Other SQLJ statements declare Java types. For example, you can declare an iterator (a construct related to a database cursor) for queries that retrieve many values, as follows:

```
#sql iterator EmpIter (String EmpNam, int EmpNumb);
```

**See Also:** For more examples and details on Oracle SQLJ syntax:

- *Oracle Database JPublisher User's Guide*
- Sample SQLJ code available on the Oracle Technology Network Web site: `http://www.oracle.com/technology/`

### Benefits of SQLJ

Oracle SQLJ extensions to Java allow rapid development and easy maintenance of applications that perform database operations through embedded SQL.

In particular, Oracle SQLJ does the following:

- Provides a concise, legible mechanism for database access from static SQL. Most SQL in applications is static. SQLJ provides more concise and less error-prone static SQL constructs than JDBC does.
- Provides an SQL Checker module for verification of syntax and semantics at translate time.
- Provides flexible deployment configurations, which makes it possible to implement SQLJ on the client, server, or middle tier.
- Supports a software standard. SQLJ is an effort of a group of vendors and will be supported by all of them. Applications can access multiple database vendors.
- Provides source code portability. Executables can be used with all of the vendor DBMSs if the code does not rely on any vendor-specific features.
- Enforces a uniform programming style for the clients and the servers.
- Integrates the SQLJ translator with **Oracle JDeveloper**, a graphical IDE that provides SQLJ translation, Java compilation, profile customizing, and debugging at the source code level, all in one step.
- Includes Oracle type extensions. Datatypes supported include: LOB datatypes, `ROWID`, `REF CURSOR`, `VARRAY`, nested table, user-defined object types, `RAW`, and `NUMBER`.

### Comparing SQLJ with JDBC

JDBC provides a complete dynamic SQL interface from Java to databases. It gives developers full control over database operations. SQLJ simplifies Java database programming to improve development productivity.

JDBC provides fine-grained control of the execution of dynamic SQL from Java, whereas SQLJ provides a higher-level binding to SQL operations in a specific database schema. Following are some differences between JDBC and SQLJ:

- SQLJ source code is more concise than equivalent JDBC source code.
- SQLJ uses database connections to type-check static SQL code. JDBC, being a completely dynamic API, does not.
- SQLJ provides strong typing of query outputs and return parameters and allows type-checking on calls. JDBC passes values to and from SQL without compile-time type checking.
- SQLJ programs allow direct embedding of Java bind expressions within SQL statements. JDBC requires a separate get or set statement for each bind variable and specifies the binding by position number.
- SQLJ provides simplified rules for calling SQL stored procedures and functions. For example, the following JDBC excerpt requires a generic call to a stored procedure or function, in this case *fun*, to have the following syntax. (This example shows SQL92 and Oracle JDBC syntaxes. Both are allowed.)

```
prepStmt.prepareCall("{call fun(?,?)}");       //stored procedure SQL92
prepStmt.prepareCall("{? = call fun(?,?)}");   //stored function SQL92
prepStmt.prepareCall("begin fun(:1,:2);end;"); //stored procedure Oracle
prepStmt.prepareCall("begin :1 := fun(:2,:3);end;");//stored func Oracle
```

Following is the SQLJ equivalent:

```
#sql {call fun(param_list) };  //Stored procedure
// Declare x
...
#sql x = {VALUES(fun(param_list)) };  // Stored function
// where VALUES is the SQL construct
```

The following benefits are common to SQLJ and JDBC:

- SQLJ source files can contain JDBC calls. SQLJ and JDBC are interoperable.
- Oracle JPublisher generates custom Java classes to be used in your SQLJ or JDBC application for mappings to Oracle object types and collections.
- Java and PL/SQL stored procedures can be used interchangeably.

### SQLJ Stored Procedures in the Server

SQLJ applications can be stored and executed in the server by using the following techniques:

- Translate, compile, and customize the SQLJ source code on a client and load the generated classes and resources into the server with the `loadjava` utility. The classes are typically stored in a Java archive (`.jar`) file.
- Load the SQLJ source code into the server, also using `loadjava`, where it is translated and compiled by the server's embedded translator.

> **See Also:** *Oracle Database JPublisher User's Guide* for more information on using stored procedures with Oracle SQLJ

## Overview of Oracle JPublisher

**Oracle JPublisher** is a code generator that automates the process of creating database-centric Java classes by hand. Oracle JPublisher is a client-side utility and is built into the database system. You can run Oracle JPublisher from the command line or directly from the Oracle JDeveloper IDE.

Oracle JPublisher inspects PL/SQL packages and database object types such as SQL object types, `VARRAY` types, and nested table types, and then generates a Java class that is a wrapper around the PL/SQL package with corresponding fields and methods.

The generated Java class can be incorporated and used by Java clients or J2EE components to exchange and transfer object type instances to and from the database transparently.

> **See Also:** *Oracle Database JPublisher User's Guide*

## Overview of Java Stored Procedures

Java stored procedures enable you to implement programs that run in the database server and which are independent of programs that run in the middle tier. Structuring applications in this way reduces complexity and increases reuse, security, performance, and scalability.

For example, you can create a Java stored procedure that performs operations that require data persistence and a separate program to perform presentation or business logic operations.

Java stored procedures interface with SQL by using a similar execution model as PL/SQL.

**See Also:** *Oracle Database Java Developer's Guide*

## Overview of Oracle Database Web Services

Web services represent a distributed computing paradigm for Java application development that is an alternative to earlier Java protocols such as JDBC. It allows application-to-application interaction by means of the XML and Web protocols. For example, an electronics parts vendor can provide a Web-based programmatic interface to its suppliers for inventory management. The vendor can call a Web service as part of a program and automatically order new stock based on the data returned.

The key technologies used in Web services are:

- Web Services Description Language (WSDL), which is a standard format for creating an XML document. WSDL describes what a web service can do, where it resides, and how to invoke it. Specifically, it describes the operations and parameters, including parameter types, provided by a Web service. In addition, a WSDL document describes the location, the transport protocol, and the invocation style for the Web service.
- Simple Object Access Protocol (SOAP) messaging, which is an XML-based message protocol used by Web services. SOAP does not prescribe a specific transport mechanism such as HTTP, FTP, SMTP, or JMS; however, most Web services accept messages that use HTTP or HTTPS.
- Universal Description, Discovery, and Integration (UDDI) business registry, which is a directory that lists Web services on the internet. The UDDI registry is often compared to a telephone directory, listing unique identifiers (white pages), business categories (yellow pages), and instructions for binding to a service protocol (green pages).

Web services can use a variety of techniques and protocols. For example:

- Dispatching can occur in a synchronous (typical) or asynchronous manner.
- You can perform invocation in an RPC-style operation in which arguments are sent and a response returned, or in a message style such as a one-way SOAP document exchange.
- You can use different encoding rules: literal or encoded.

You can invoke a Web service statically, in which case you may know everything about it beforehand, or dynamically, in which case you can discover its operations and transport endpoints on the fly.

### Oracle Database as a Web Service Provider

Oracle Database can function as either a Web service provider or as a Web service consumer. When used as a provider, the database enables sharing and disconnected access to stored procedures, data, metadata, and other database resources such as the queuing and messaging systems.

As a Web service provider, Oracle Database provides a disconnected and heterogeneous environment that:

- Exposes stored procedures independently of the language in which the procedures are written
- Exposes SQL Queries and XQuery

## Overview of Writing Procedures and Functions in Java

Functions and procedures are named blocks that encapsulate a sequence of statements. They are like building blocks that you can use to construct modular, maintainable applications. You write these named blocks and then define them by using the `loadjava` command or SQL `CREATE FUNCTION`, `CREATE PROCEDURE`, or `CREATE PACKAGE` statements. These Java methods can accept arguments and are callable from:

- SQL `CALL` statements
- Embedded SQL `CALL` statements
- PL/SQL blocks, subprograms, and packages
- DML statements (`INSERT`, `UPDATE`, `DELETE`, and `SELECT`)
- Oracle development tools such as OCI, Pro*C/C++, and Oracle Developer
- Oracle Java interfaces such as JDBC, SQLJ statements, CORBA, and Enterprise Java Beans
- Method calls from object types

### Overview of Writing Database Triggers in Java

A database trigger is a stored procedure that Oracle Database invokes ("fires") automatically when certain events occur, for example, when a DML operation modifies a certain table. Triggers enforce business rules, prevent incorrect values from being stored, and reduce the need to perform checking and cleanup operations in each application.

### Why Use Java for Stored Procedures and Triggers?

- Stored procedures and triggers are compiled once, are easy to use and maintain, and require less memory and computing overhead.
- Network bottlenecks are avoided, and response time is improved. Distributed applications are easier to build and use.
- Computation-bound procedures run faster in the server.
- Data access can be controlled by letting users have only stored procedures and triggers that execute with their definer's privileges instead of invoker's rights.
- PL/SQL and Java stored procedures can call each other.
- Java in the server follows the Java language specification and can use the SQLJ standard, so that databases other than Oracle Database are also supported.
- Stored procedures and triggers can be reused in different applications as well as different geographic sites.

# Overview of Pro*C/C++

The Pro*C/C++ precompiler is a software tool that allows the programmer to embed SQL statements in a C or C++ source file. Pro*C/C++ reads the source file as input and outputs a C or C++ source file that replaces the embedded SQL statements with Oracle runtime library calls and is then compiled by the C or C++ compiler.

When there are errors found during the precompilation or the subsequent compilation, modify your precompiler input file and re-run the two steps.

## Implementing a Pro*C/C++ Application

The following is a simple code fragment from a C source file that queries the table `employees` in the schema `hr`:

```
...
#define  UNAME_LEN   10
...
int   emp_number;
/* Define a host structure for the output values of a SELECT statement. */
/* No declare section needed if precompiler option MODE=ORACLE          */
struct {
    VARCHAR  last_name[UNAME_LEN];
    float    salary;
    float    commission_pct;
} emprec;
/* Define an indicator structure to correspond to the host output structure. */
struct {
    short emp_name_ind;
    short sal_ind;
    short comm_ind;
} emprec_ind;
...
/* Select columns last_name, salary, and commission_pct given the user's input
/* for employee_id. */
    EXEC SQL SELECT last_name, salary, commission_pct
        INTO :emprec INDICATOR :emprec_ind
        FROM employees
        WHERE employee_id = :emp_number;
...
```

The embedded `SELECT` statement is only slightly different from an interactive (SQL*Plus) `SELECT` statement. Every embedded SQL statement begins with EXEC SQL. The colon (:), precedes every host (C) variable. The returned values of data and indicators (set when the data value is `NULL` or character columns have been truncated) can be stored in structs (such as in the preceding code fragment), in arrays, or in arrays of structs. Multiple result set values are handled very simply in a manner that resembles the case shown, where there is only one result, because of the unique employee number. You use the actual names of columns and tables in embedded SQL.

Use the default precompiler option values, or you can enter values which give you control over the use of resources, how errors are reported, the formatting of output, and how cursors (which correspond to a particular connection or SQL statement) are managed. Cursors are used when there are multiple result set values.

Enter the options either in a configuration file, on the command line, or in-line inside your source code with a special statement that begins with `EXEC ORACLE`. If there are no errors found, you can then compile, link, and execute the output source file, like any other C program that you write.

Use the precompiler to create server database access from clients that can be on many different platforms. Pro*C/C++ allows you the freedom to design your own user interfaces and to add database access to existing applications.

Before writing your embedded SQL statements, you may want to test interactive versions of the SQL in SQL*Plus. You then make only minor changes to start testing your embedded SQL application.

## Highlights of Pro*C/C++ Features

The following is a short subset of the capabilities of Pro*C/C++. For complete details, refer to the *Pro*C/C++ Precompiler Programmer's Guide*.

- You can write your application in either C or C++.
- You can write multithreaded programs if your platform supports a threads package. Concurrent connections are supported in either single-threaded or multithreaded applications.
- You can improve performance by embedding PL/SQL blocks. These blocks can call functions or procedures in Java or PL/SQL that are written by you or provided in Oracle Database packages.
- Using precompiler options, you can check the syntax and semantics of your SQL or PL/SQL statements during precompilation, as well as at runtime.
- You can call stored PL/SQL and Java subprograms. Modules written in COBOL or in C can be called from Pro*C/C++. External C procedures in shared libraries are callable by your program.
- You can conditionally precompile sections of your code so that they can execute in different environments.
- You can use arrays, or structures, or arrays of structures as host and indicator variables in your code to improve performance.
- You can deal with errors and warnings so that data integrity is guaranteed. As a programmer, you control how errors are handled.
- Your program can convert between internal datatypes and C language datatypes.
- The Oracle Call Interface (OCI) and Oracle C++ Call Interface (OCCI), lower-level C and C++ interfaces, are available for use in your precompiler source.
- Pro*C/C++ supports dynamic SQL, a technique that allows users to input variable values and statement syntax.
- Pro*C/C++ can use special SQL statements to manipulate tables containing user-defined object types. An Object Type Translator (OTT) will map the object types and named collection types in your database to structures and headers that you will then include in your source.
- Two kinds of collection types, nested tables and `VARRAY`, are supported with a set of SQL statements that allow a high degree of control over data.
- Large Objects are accessed by another set of SQL statements.
- A new ANSI SQL standard for dynamic SQL is supported for new applications, so that you can execute SQL statements with a varying number of host variables. An older technique for dynamic SQL is still usable by pre-existing applications.
- Globalization support lets you use multibyte characters and UCS2 Unicode data.
- Using scrollable cursors, you can move backward and forward through a result set. For example, you can fetch the last row of the result set, or jump forward or backward to an absolute or relative position within the result set.
- A connection pool is a group of physical connections to a database that can be shared by several named connections. Enabling the connection pool option can help to optimize the performance of Pro*C/C++ application. The connection pool option is not enabled by default.

# Overview of Pro*COBOL

The Pro*COBOL precompiler is a software tool that allows the programmer to embed SQL statements in a COBOL source code file. Pro*COBOL reads the source file as input and outputs a COBOL source file that replaces the embedded SQL statements with Oracle Database runtime library calls, and is then compiled by the COBOL compiler.

When there are errors found during the precompilation or the subsequent compilation, modify your precompiler input file and re-run the two steps.

## How You Implement a Pro*COBOL Application

Here is a simple code fragment from a source file that queries the table `employees` in the schema `hr`:

```
...
 WORKING-STORAGE SECTION.
*
* DEFINE HOST INPUT AND OUTPUT HOST AND INDICATOR VARIABLES.
* NO DECLARE SECTION NEEDED IF MODE=ORACLE.
*
 01  EMP-REC-VARS.
     05  EMP-NAME    PIC X(10) VARYING.
     05  EMP-NUMBER  PIC S9(4) COMP VALUE ZERO.
     05  SALARY      PIC S9(5)V99 COMP-3 VALUE ZERO.
     05  COMMISSION  PIC S9(5)V99 COMP-3 VALUE ZERO.
     05  COMM-IND    PIC S9(4) COMP VALUE ZERO.
...
 PROCEDURE DIVISION.
...
     EXEC SQL
         SELECT last_name, salary, commission_pct
         INTO :EMP-NAME, :SALARY, :COMMISSION:COMM-IND
         FROM employees
         WHERE employee_id = :EMP-NUMBER
     END-EXEC.
...
```

The embedded `SELECT` statement is only slightly different from an interactive (SQL*Plus) `SELECT` statement. Every embedded SQL statement begins with EXEC SQL. The colon (:) precedes every host (COBOL) variable. The SQL statement is terminated by END-EXEC. The returned values of data and indicators (set when the data value is `NULL` or character columns have been truncated) can be stored in group items (such as in the preceding code fragment), in tables, or in tables of group items. Multiple result set values are handled very simply in a manner that resembles the case shown, where there is only one result, given the unique employee number. You use the actual names of columns and tables in embedded SQL.

Use the default precompiler option values, or enter values that give you control over the use of resources, how errors are reported, the formatting of output, and how cursors are managed (cursors correspond to a particular connection or SQL statement).

Enter the options in a configuration file, on the command line, or in-line inside your source code with a special statement that begins with EXEC ORACLE. If there are no errors found, you can then compile, link, and execute the output source file, like any other COBOL program that you write.

Use the precompiler to create server database access from clients that can be on many different platforms. Pro*COBOL allows you the freedom to design your own user interfaces and to add database access to existing COBOL applications.

The embedded SQL statements available conform to an ANSI standard, so that you can access data from many databases in a program, including remote servers networked through Oracle Net.

Before writing your embedded SQL statements, you may want to test interactive versions of the SQL in SQL*Plus. You then make only minor changes to start testing your embedded SQL application.

### Highlights of Pro*COBOL Features

The following is a short subset of the capabilities of Pro*COBOL.

- You can call stored PL/SQL or Java subprograms. You can improve performance by embedding PL/SQL blocks. These blocks can call PL/SQL functions or procedures written by you or provided in Oracle Database packages.
- Precompiler options allow you to define how cursors, errors, syntax-checking, file formats, and so on, are handled.
- Using precompiler options, you can check the syntax and semantics of your SQL or PL/SQL statements during precompilation, as well as at runtime.
- You can conditionally precompile sections of your code so that they can execute in different environments.
- Use tables, or group items, or tables of group items as host and indicator variables in your code to improve performance.
- You can program how errors and warnings are handled, so that data integrity is guaranteed.
- Pro*COBOL supports dynamic SQL, a technique that allows users to input variable values and statement syntax.

**See Also:** *Pro*COBOL Programmer's Guide* for complete details

## Overview of OCI and OCCI

The Oracle Call Interface (OCI) and Oracle C++ Call Interface (OCCI) are application programming interfaces (APIs) that allow you to create applications that use native procedures or function calls of a third-generation language to access Oracle Database and control all phases of SQL statement execution. These APIs provide:

- Improved performance and scalability through the efficient use of system memory and network connectivity
- Consistent interfaces for dynamic session and transaction management in a two-tier client/server or multitier environment
- N-tiered authentication
- Comprehensive support for application development using Oracle objects
- Access to external databases
- Ability to develop applications that service an increasing number of users and requests without additional hardware investments

OCI lets you manipulate data and schemas in a database using a host programming language, such as C. OCCI is an object-oriented interface suitable for use with C++. These APIs provide a library of standard database access and retrieval functions in the form of a dynamic runtime library (OCILIB) that can be linked in an application at runtime. This eliminates the need to embed SQL or PL/SQL within 3GL programs.

**See Also:** For more information about OCI and OCCI calls:

- *Oracle Call Interface Programmer's Guide*
- *Oracle C++ Call Interface Programmer's Guide*
- *Oracle Streams Advanced Queuing User's Guide and Reference*
- *Oracle Database Globalization Support Guide*
- *Oracle Database Data Cartridge Developer's Guide*

## Advantages of OCI

OCI provides significant advantages over other methods of accessing Oracle Database:

- More fine-grained control over all aspects of the application design.
- High degree of control over program execution.
- Use of familiar 3GL programming techniques and application development tools such as browsers and debuggers.
- Support of dynamic SQL, method 4.
- Availability on the broadest range of platforms of all the Oracle programmatic interfaces.
- Dynamic bind and define using callbacks.
- Describe functionality to expose layers of server metadata.
- Asynchronous event notification for registered client applications.
- Enhanced array data manipulation language (DML) capability for array `INSERT`s, `UPDATE`s, and `DELETE`s.
- Ability to associate a commit request with an execute to reduce round-trips.
- Optimization for queries using transparent prefetch buffers to reduce round-trips.
- Thread safety, so you do not have to implement mutual exclusion (mutex) locks on OCI handles.
- The server connection in nonblocking mode means that control returns to the OCI code when a call is still executing or could not complete.

## Parts of the OCI

The OCI encompasses four main sets of functionality:

- OCI *relational functions*, for managing database access and processing SQL statements
- OCI *navigational functions*, for manipulating objects retrieved from an Oracle Database
- OCI *datatype mapping and manipulation functions*, for manipulating data attributes of Oracle types
- OCI *external procedure functions*, for writing C callbacks from PL/SQL

## Procedural and Non-Procedural Elements

The Oracle Call Interface (OCI) enables you to develop applications that combine the non-procedural data access power of SQL with the procedural capabilities of most programming languages, including C and C++. Note the following characteristics of procedural and non-procedural languages:

- In a non-procedural language program, the set of data to be operated on is specified, but what operations will be performed and how the operations are to be carried out is not specified. The non-procedural nature of SQL makes it an easy language to learn and to use to perform database transactions. It is also the standard language used to access and manipulate data in modern relational and object-relational database systems.
- In a procedural language program, the execution of most statements depends on previous or subsequent statements and on control structures, such as loops or conditional branches, which are not available in SQL. The procedural nature of these languages makes them more complex than SQL, but it also makes them very flexible and powerful.

The combination of both non-procedural and procedural language elements in an OCI program provides easy access to Oracle Database in a structured programming environment.

The OCI supports all SQL data definition, data manipulation, query, and transaction control facilities that are available through Oracle Database. For example, an OCI program can run a query against Oracle Database. The queries can require the program to supply data to the database using input (bind) variables, as follows:

```
SELECT name FROM employees WHERE empno = :empnumber
```

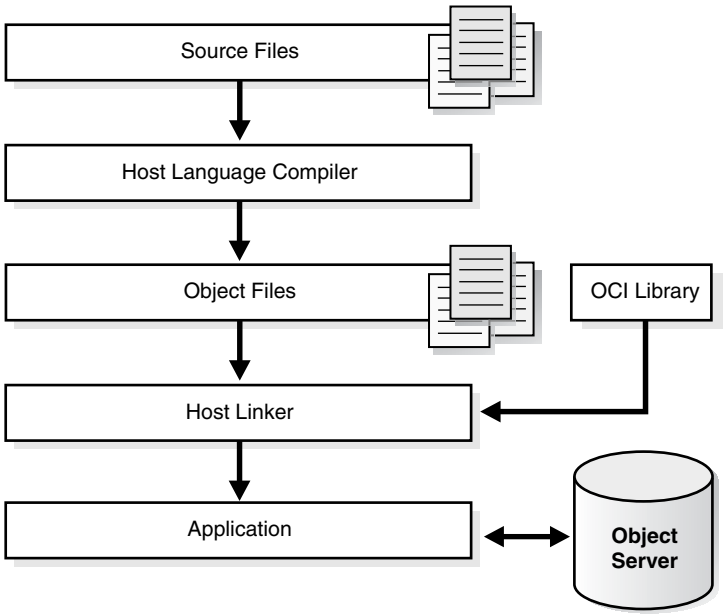
In the preceding SQL statement, `:empnumber` is a placeholder for a value that will be supplied by the application.

You can alternatively use PL/SQL, Oracle's procedural extension to SQL. The applications you develop can be more powerful and flexible than applications written in SQL alone. The OCI also provides facilities for accessing and manipulating objects in Oracle Database.

## Building an OCI Application

As Figure 1–1 shows, you compile and link an OCI program in the same way that you compile and link a non-database application. There is no need for a separate preprocessing or precompilation step.

***Figure 1–1 The OCI Development Process***




**Note:** To properly link your OCI programs, it may be necessary on some platforms to include other libraries, in addition to the OCI library. Check your Oracle platform-specific documentation for further information about extra libraries that may be required.

# Overview of Oracle Data Provider for .NET (ODP.NET)

Oracle Data Provider for .NET (ODP.NET) is an implementation of a data provider for Oracle Database.

ODP.NET uses APIs native to Oracle Database to offer fast and reliable access from any .NET application to database features and data. It also uses and inherits classes and interfaces available in the Microsoft .NET Framework Class Library.

For programmers using Oracle Provider for OLE DB, ADO (ActiveX Data Objects) provides an automation layer that exposes an easy programming model. ADO.NET provides a similar programming model, but without the automation layer, for better performance. More importantly, the ADO.NET model allows native providers such as ODP.NET to expose specific features and datatypes specific to Oracle Database.

**See Also:** *Oracle Data Provider for .NET Developer's Guide*

## Using ODP.NET in a Simple Application

The following is a simple C# application that connects to Oracle Database and displays its version number before disconnecting.

```
using System;
using Oracle.DataAccess.Client;

class Example
{
  OracleConnection con;

  void Connect()
```

```
  {
    con = new OracleConnection();
    con.ConnectionString = "User Id=hr;Password=hr;Data Source=oracle";
    con.Open();
    Console.WriteLine("Connected to Oracle" + con.ServerVersion);
  }

  void Close()
  {
    con.Close();
    con.Dispose();
  }

  static void Main()
  {
    Example example = new Example();
    example.Connect();
    example.Close();
  }
}
```

**Note:** Additional samples are provided in directory *ORACLE_BASE**ORACLE_HOME*\ODP.NET\Samples.

## Overview of Oracle Objects for OLE (OO4O)

Oracle Objects for OLE (OO4O) is a product designed to allow easy access to data stored in Oracle Database with any programming or scripting language that supports the Microsoft COM Automation and ActiveX technology. This includes Visual Basic, Visual C++, Visual Basic For Applications (VBA), IIS Active Server Pages (VBScript and JavaScript), and others.

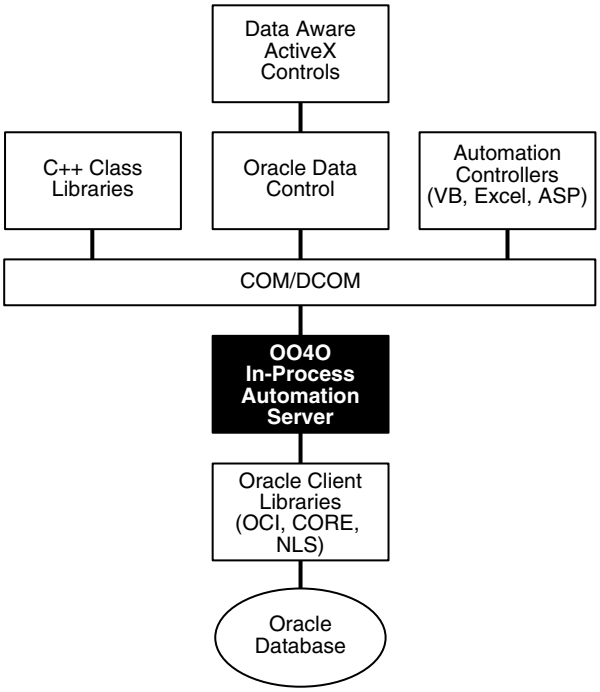
See the OO4O online help for detailed information about using OO4O.

Oracle Objects for OLE consists of the following software layers:

- OO4O "In-Process" Automation Server
- Oracle Data Control
- Oracle Objects for OLE C++ Class Library

Figure 1–2, "Software Layers" illustrates the OO4O software components.

*Figure 1–2 Software Layers*




## OO4O Automation Server

The OO4O Automation Server is a set of COM Automation objects for connecting to Oracle Database, executing SQL statements and PL/SQL blocks, and accessing the results.

Unlike other COM-based database connectivity APIs, such as Microsoft ADO, the OO4O Automation Server has been developed and evolved specifically for use with Oracle Database.

It provides an optimized API for accessing features that are unique to Oracle Database and are otherwise cumbersome or inefficient to use from ODBC or OLE database-specific components.

OO4O provides key features for accessing Oracle Database efficiently and easily in environments ranging from the typical two-tier client/server applications, such as those developed in Visual Basic or Excel, to application servers deployed in multitiered application server environments such as Web server applications in Microsoft Internet Information Server (IIS) or Microsoft Transaction Server (MTS).

Features include:

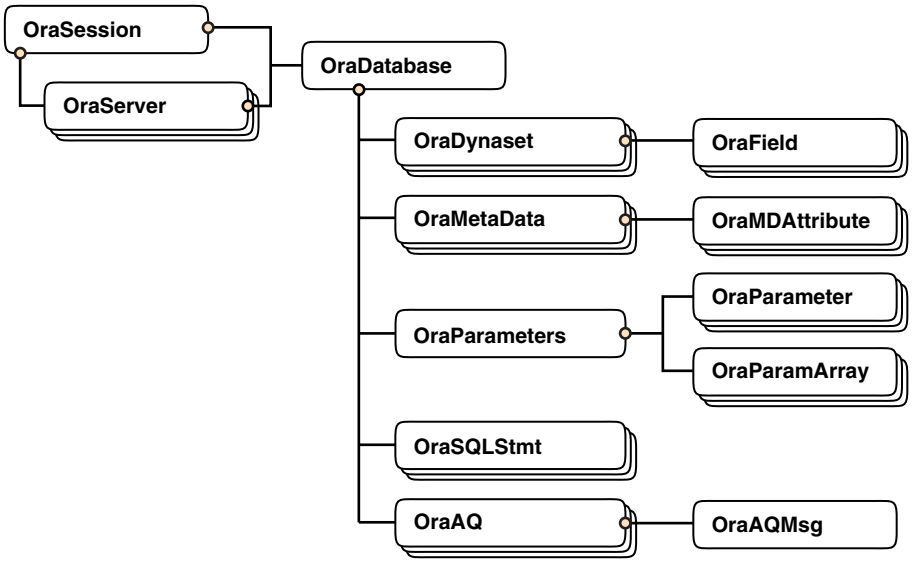
- Support for execution of PL/SQL and Java stored procedures, and PL/SQL anonymous blocks. This includes support for Oracle datatypes used as parameters to stored procedures, including PL/SQL cursors. Refer to "Support for Oracle LOB and Object Datatypes" on page 1-28.
- Support for scrollable and updatable cursors for easy and efficient access to result sets of queries.
- Thread-safe objects and Connection Pool Management Facility for developing efficient Web server applications.
- Full support for Oracle object-relational and LOB datatypes.

- Full support for Advanced Queuing.
- Support for array inserts and updates.
- Support for Microsoft Transaction Server (MTS).

## OO4O Object Model

The Oracle Objects for OLE object model is illustrated in Figure 1–3, "Objects and Their Relations".

***Figure 1–3 Objects and Their Relations***




### OraSession

An OraSession object manages collections of OraDatabase, OraConnection, and OraDynaset objects used within an application.

Typically, a single OraSession object is created for each application, but you can create named OraSession objects for shared use within and between applications.

The OraSession object is the top-most object for an application. It is the only object created by the CreateObject VB/VBA API and not by an Oracle Objects for OLE method. The following code fragment shows how to create an OraSession object:

```
Dim OraSession as Object
Set OraSession = CreateObject("OracleInProcServer.XOraSession")
```

### OraServer

OraServer represents a physical network connection to Oracle Database.

The `OraServer` interface is introduced to expose the connection-multiplexing feature provided in the Oracle Call Interface. After an `OraServer` object is created, multiple user sessions (`OraDatabase`) can be attached to it by invoking the `OpenDatabase` method. This feature is particularly useful for application components, such as Internet Information Server (IIS), that use Oracle Objects for OLE in n-tier distributed environments.

The use of connection multiplexing when accessing Oracle Database with a large number of user sessions active can help reduce server processing and resource requirements while improving server scalability.

OraServer is used to share a single connection across multiple OraDatabase objects (multiplexing), whereas each OraDatabase obtained from an OraSession has its own physical connection.

### OraDatabase

An `OraDatabase` interface adds additional methods for controlling transactions and creating interfaces representing of Oracle object types. Attributes of schema objects can be retrieved using the `Describe` method of the `OraDatabase` interface.

In releases prior to Oracle8*i*, an `OraDatabase` object is created by invoking the `OpenDatabase` method of an `OraSession` interface. The Oracle Net alias, user name, and password are passed as arguments to this method. In Oracle8*i* and later, invocation of this method results in implicit creation of an `OraServer` object.

An `OraDatabase` object can also be created using the `OpenDatabase` method of the `OraServer` interface.

Transaction control methods are available at the `OraDatabase` (user session) level. Transactions may be started as `Read-Write` (default), `Serializable`, or `Read-only`. Transaction control methods include:

- `BeginTrans`
- `CommitTrans`
- `RollbackTrans`

For example:

```
UserSession.BeginTrans(OO4O_TXN_READ_WRITE)
UserSession.ExecuteSQL("delete emp where empno = 1234")
UserSession.CommitTrans
```

### OraDynaset

An `OraDynaset` object permits browsing and updating of data created from a SQL `SELECT` statement.

The `OraDynaset` object can be thought of as a cursor, although in actuality several real cursors may be used to implement the semantics of `OraDynaset`. An `OraDynaset` object automatically maintains a local cache of data fetched from the server and transparently implements scrollable cursors within the browse data. Large queries may require significant local disk space; application developers are encouraged to refine queries to limit disk usage.

### OraField

An `OraField` object represents a single column or data item within a row of a dynaset.

If the current row is being updated, then the `OraField` object represents the currently updated value, although the value may not yet have been committed to the database.

Assignment to the `Value` property of a field is permitted only if a record is being edited (using `Edit`) or a new record is being added (using `AddNew`). Other attempts to assign data to a field's `Value` property results in an error.

### OraMetaData and OraMDAttribute

An `OraMetaData` object is a collection of `OraMDAttribute` objects that represent the description information about a particular schema object in the database.

The `OraMetaData` object can be visualized as a table with three columns:

- `Metadata Attribute Name`
- `Metadata Attribute Value`
- `Flag` specifying whether the `Value` is another `OraMetaData` object

The `OraMDAttribute` objects contained in the `OraMetaData` object can be accessed by subscripting using ordinal integers or by using the name of the property. Referencing a subscript that is not in the collection results in the return of a `NULL` `OraMDAttribute` object.

### OraParameters and OraParameter

An `OraParameter` object represents a bind variable in a SQL statement or PL/SQL block.

`OraParameter` objects are created, accessed, and removed indirectly through the `OraParameters` collection of an `OraDatabase` object. Each parameter has an identifying name and an associated value. You can automatically bind a parameter to SQL and PL/SQL statements of other objects (as noted in the object descriptions), by using the parameter name as a placeholder in the SQL or PL/SQL statement. Such use of parameters can simplify dynamic queries and increase program performance.

### OraParamArray

An `OraParamArray` object represents an array-type bind variable in a SQL statement or PL/SQL block, as opposed to a scalar-type bind variable represented by the `OraParameter` object.

`OraParamArray` objects are created, accessed, and removed indirectly through the `OraParameters` collection of an `OraDatabase` object. Each `OraParamArray` object has an identifying name and an associated value.

### OraSQLStmt

An `OraSQLStmt` object represents a single SQL statement. Use the `CreateSQL` method to create an `OraSQLStmt` object from an `OraDatabase` object.

During create and refresh, `OraSQLStmt` objects automatically bind all relevant, enabled input parameters to the specified SQL statement, using the parameter names as placeholders in the SQL statement. This can improve the performance of SQL statement execution without re-parsing the SQL statement.

The `OraSQLStmt` object can be used later to execute the same query using a different value for the `:SALARY` placeholder. This is done as follows (`updateStmt` is the `OraSQLStmt` object here):

```
OraDatabase.Parameters("SALARY").value = 200000
updateStmt.Parameters("ENAME").value = "KING"
updateStmt.Refresh
```

### OraAQ

An `OraAQ` object is instantiated by invoking the `CreateAQ` method of the `OraDatabase` interface. It represents a queue that is present in the database.

Oracle Objects for OLE provides interfaces for accessing Oracle Advanced Queuing (AQ) feature. It makes AQ accessible from popular COM-based development environments such as Visual Basic. For a detailed description of Oracle Advanced Queuing, refer to *Oracle Streams Advanced Queuing User's Guide and Reference*.

### OraAQMsg

The `OraAQMsg` object encapsulates the message to be enqueued or dequeued. The message can be of any user-defined or raw type.

For a detailed description of Oracle Advanced Queuing, refer to *Oracle Streams Advanced Queuing User's Guide and Reference.*

### OraAQAgent

The `OraAQAgent` object represents a message recipient and is only valid for queues that allow multiple consumers. It is a child of `OraAQMsg`.

An `OraAQAgent` object can be instantiated by invoking the `AQAgent` method. For example:

```
Set agent = qMsg.AQAgent(name)
```

An `OraAQAgent` object can also be instantiated by invoking the `AddRecipient` method. For example:

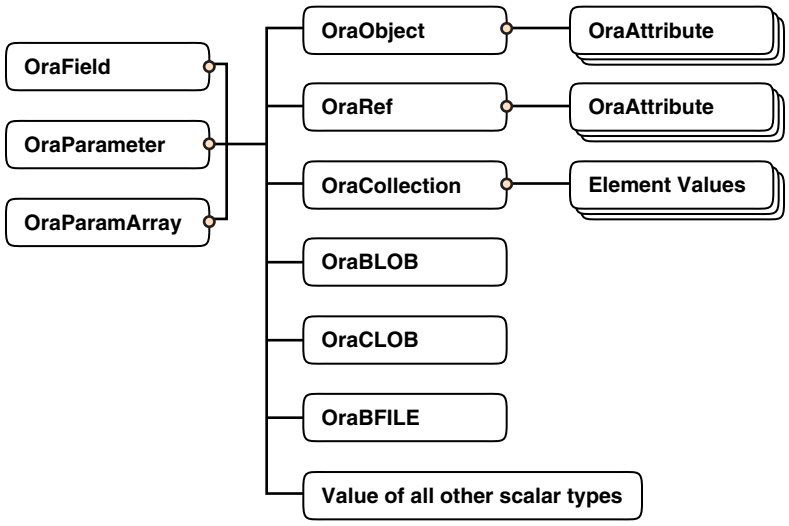
```
Set agent = qMsg.AddRecipient(name, address, protocol).
```

## Support for Oracle LOB and Object Datatypes

Oracle Objects for OLE provides full support for accessing and manipulating instances of object datatypes and LOBs in Oracle Database. Figure 1–4, "Supported Oracle Datatypes" illustrates the datatypes supported by OO4O.

Instances of these types can be fetched from the database or passed as input or output variables to SQL statements and PL/SQL blocks, including stored procedures and functions. All instances are mapped to COM Automation Interfaces that provide methods for dynamic attribute access and manipulation.

***Figure 1–4 Supported Oracle Datatypes***



### OraBLOB and OraCLOB

The `OraBlob` and `OraClob` interfaces in Oracle Objects for OLE provide methods for performing operations on large database objects of datatype `BLOB`, `CLOB`, and `NCLOB`. `BLOB`, `CLOB`, and `NCLOB` datatypes are also referred to here as **LOB** datatypes.

LOB data is accessed using `Read` and the `CopyToFile` methods.

LOB data is modified using `Write`, `Append`, `Erase`, `Trim`, `Copy`, `CopyFromFile`, and `CopyFromBFile` methods. Before modifying the content of a LOB column in a row, a row lock must be obtained. If the LOB column is a field of an `OraDynaset`, object, then the lock is obtained by invoking the `Edit` method.

### OraBFILE

The `OraBFile` interface in Oracle Objects for OLE provides methods for performing operations on large database objects of datatype `BFILE`.

`BFILE` objects are large binary data objects stored in operating system files outside of the database tablespaces.

## Oracle Data Control

Oracle Data Control (ODC) is an ActiveX Control that is designed to simplify the exchange of data between Oracle Database and visual controls such edit, text, list, and grid controls in Visual Basic and other development tools that support custom controls.

ODC acts as an agent to handle the flow of information from Oracle Database and a visual data-aware control, such as a grid control, that is bound to it. The data control manages various user interface (UI) tasks such as displaying and editing data. It also executes and manages the results of database queries.

Oracle Data Control is compatible with the Microsoft data control included with Visual Basic. If you are familiar with the Visual Basic data control, learning to use Oracle Data Control is quick and easy. Communication between data-aware controls and a Data Control is governed by a protocol that has been specified by Microsoft.

## Oracle Objects for OLE C++ Class Library

Oracle Objects for OLE C++ Class Library is a collection of C++ classes that provide programmatic access to the Oracle Object Server. Although the class library is implemented using OLE Automation, neither the OLE development kit nor any OLE development knowledge is necessary to use it. This library helps C++ developers avoid the chore of writing COM client code for accessing the OO4O interfaces.

## Additional Sources of Information

For detailed information about Oracle Objects for OLE refer to the online help provided with the OO4O product:

- Oracle Objects for OLE Help
- Oracle Objects for OLE C++ Class Library Help

To view examples of how to use Oracle Objects for OLE, refer to the samples located in the `ORACLE_HOME\OO4O` directory of the Oracle Database installation. Additional OO4O examples can be found in the following Oracle publications:

- *Oracle Database Application Developer's Guide - Large Objects*

- *Oracle Streams Advanced Queuing User's Guide and Reference*
- *Oracle Database PL/SQL Packages and Types Reference*

# Choosing a Programming Environment

To choose a programming environment for a new development project:

- Review the preceding overviews and the manuals for each environment.
- Read the platform-specific manual that explains which compilers are approved for use with your platforms.
- If a particular language does not provide a feature you need, remember that PL/SQL and Java stored procedures can both be called from code written in any of the languages in this chapter. Stored procedures include triggers and object type methods.
- External procedures written in C can be called from OCI, Java, PL/SQL or SQL. The external procedure itself can call back into the database using either SQL, OCI, or Pro*C (but not C++).

The following examples illustrate easy choices:

- Pro*COBOL does not support object types or collection types, while Pro*C/C++ does.
- SQLJ does not support dynamic SQL the way that JDBC does.

## Choosing Whether to Use OCI or a Precompiler

Precompiler applications typically contain less code than equivalent OCI applications, which can help productivity.

Some situations require detailed control of the database and are suited for OCI applications (either pure OCI or a precompiler application with embedded OCI calls):

- OCI provides more detailed control over multiplexing and migrating sessions.
- OCI provides dynamic bind and define using callbacks that can be used for any arbitrary structure, including lists.
- OCI has many calls to handle metadata.
- OCI allows asynchronous event notifications to be received by a client application. It provides a means for clients to generate notifications for propagation to other clients.
- OCI allows DML statements to use arrays to complete as many iterations as possible before returning any error messages.
- OCI calls for special purposes include Advanced Queuing, globalization support, Data Cartridges, and support of the date and time datatypes.
- OCI calls can be embedded in a Pro*C/C++ application.

## Using Built-In Packages and Libraries

Both Java and PL/SQL have built-in packages and libraries.

PL/SQL and Java interoperate in the server. You can execute a PL/SQL package from Java or wrap a PL/SQL class with a Java wrapper so that it can be called from

distributed CORBA and EJB clients. The following table shows PL/SQL packages and their Java equivalents:

***Table 1–1 PL/SQL and Java Equivalent Software***

| PL/SQL Package | Java Equivalent |
|---|---|
| DBMS_ALERT | Call package with SQLJ or JDBC. |
| DBMS_DDL | JDBC has this functionality. |
| DBMS_JOB | Schedule a job that has a Java Stored procedure. |
| DBMS_LOCK | Call with SQLJ or JDBC. |
| DBMS_MAIL | Use JavaMail. |
| DBMS_OUTPUT | Use subclass oracle.aurora.rdbms.OracleDBMSOutputStream or Java stored procedure DBMS_JAVA.SET_STREAMS. |
| DBMS_PIPE | Call with SQLJ or JDBC. |
| DBMS_SESSION | Use JDBC to execute an ALTER SESSION statement. |
| DBMS_SNAPSHOT | Call with SQLJ or JDBC. |
| DBMS_SQL | Use JDBC. |
| DBMS_TRANSACTION | Use JDBC to execute an ALTER SESSION statement. |
| DBMS_UTILITY | Call with SQLJ or JDBC. |
| UTL_FILE | Grant the JAVAUSERPRIV privilege and then use Java I/O entry points. |

## Java Compared to PL/SQL

Both Java and PL/SQL can be used to build applications in the database. Here are some guidelines for their use:

### PL/SQL Is Optimized for Database Access

PL/SQL uses the same datatypes as SQL. SQL datatypes are thus easier to use and SQL operations are faster than with Java, especially when a large amount of data is involved, when mostly database access is done, or when bulk operations are used.

### PL/SQL Is Integrated with the Database

PL/SQL is an extension to SQL offering data encapsulation, information hiding, overloading, and exception-handling.

Some advanced PL/SQL capabilities are not available for Java in Oracle9*i*. Examples are autonomous transactions and the dblink facility for remote databases. Code development is usually faster in PL/SQL than in Java.

### Both Java and PL/SQL Have Object-Oriented Features

Java has inheritance, polymorphism, and component models for developing distributed systems. PL/SQL has inheritance and **type evolution**, the ability to change methods and attributes of a type while preserving subtypes and table data that use the type.

### Java Is Used for Open Distributed Applications

Java has a richer type system than PL/SQL and is an object-oriented language. Java can use CORBA (which can have many different computer languages in its clients) and EJB. PL/SQL packages can be called from CORBA or EJB clients.

You can run XML tools, the Internet File System, or JavaMail from Java.

Many Java-based development tools are available throughout the industry.

# 11

# Developing Applications with the PL/SQL Web Toolkit

Java is not the only language that can do network operations and produce dynamic Web content. PL/SQL has a number of features that you can use to make your database available on the Web and make back-office data accessible on the intranet.

This chapter discusses the following topics:


- Developing PL/SQL Web Applications: Overview
- Using the mod_plsql Gateway
- Generating HTML Output with PL/SQL
- Passing Parameters to a PL/SQL Web Application
- Performing Network Operations within PL/SQL Stored Procedures


## Developing PL/SQL Web Applications: Overview

This section contains the following topics:


- Invoking a PL/SQL Web Application
- Implementing a PL/SQL Web Application

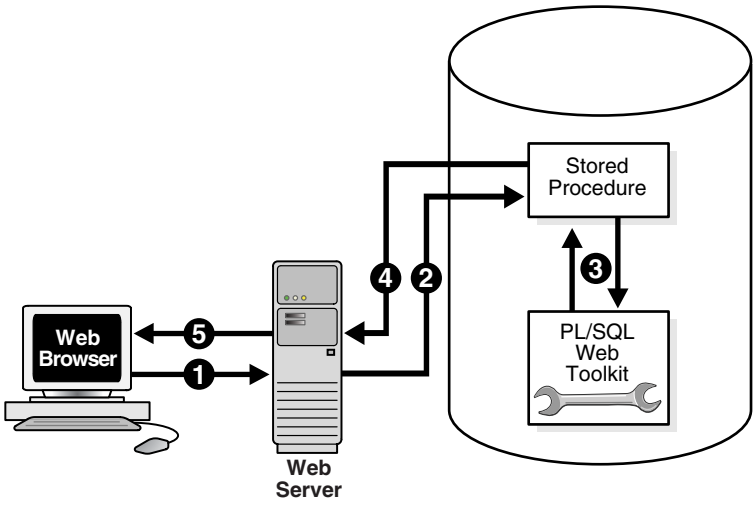

### Invoking a PL/SQL Web Application

Typically, a Web application written in PL/SQL is a set of stored procedures that interact with Web browsers through HTTP. A set of interlinked, dynamically generated HTML pages forms the user interface of a web application.

The program flow of a PL/SQL Web application is similar to that in a CGI Perl script. Developers often use CGI scripts to produce Web pages dynamically, but such scripts are often not optimal for accessing Oracle Database. Delivering Web content with PL/SQL stored procedures provides the power and flexibility of database processing. For example, you can use DML, dynamic SQL, and cursors. You also eliminate the process overhead of forking a new CGI process to handle each HTTP request.

Figure 11–1 illustrates the generic process for a PL/SQL Web application.

*Figure 11–1 PL/SQL Web Application*




The process includes the following steps:

1. A user visits a Web page, follows a hypertext link, or submits data in a form, which causes the browser to send a HTTP request for a URL to an HTTP server.
2. The HTTP server invokes a stored procedure on an Oracle database according to the data encoded in the URL. The data in the URL takes the form of parameters to be passed to the stored procedure.
3. The stored procedure calls subprograms in the PL/SQL Web Toolkit. Typically, subprograms such as `HTP.Print` generate Web pages dynamically. A generated Web page varies depending on the database contents and the input parameters.
4. The subprograms pass the dynamically generated page to the Web server.
5. The Web server delivers the page to the client.

## Implementing a PL/SQL Web Application

You can implement a Web browser-based application entirely in PL/SQL with the Oracle Database components described in this section.

### PL/SQL Web Toolkit

This set of PL/SQL packages is a generic interface that enables you to use stored procedures called by mod_plsql at runtime.

In response to a browser request, a PL/SQL procedure updates or retrieves data from Oracle Database according to the user input. It then generates an HTTP response to the browser, typically in the form of a file download or HTML to be displayed. The Web Toolkit API enables stored procedures to perform actions such as the following:

- Obtain information about an HTTP request
- Generate HTTP headers such as content-type and mime-type
- Set browser cookies
- Generate HTML pages

Table 11–1 describes commonly used PL/SQL Web Toolkit packages.

*Table 11–1 Commonly Used Packages in the PL/SQL Web Toolkit*

| Package | Description of Contents |
|---|---|
| HTF | Function versions of the procedures in the htp package. The function versions do not directly generate output in a Web page. Instead, they pass their output as return values to the statements that invoke them. Use these functions when you need to nest function calls. |
| HTP | Procedures that generate HTML tags. For instance, the procedure htp.anchor generates the HTML anchor tag, <A>. |
| OWA_CACHE | Functions and procedures that enable the PL/SQL gateway cache feature to improve performance of your PL/SQL Web application.<br>You can use this package to enable expires-based and validation-based caching with the PL/SQL gateway file system. |
| OWA_COOKIE | Subprograms that send and retrieve HTTP cookies to and from a client Web browser. Cookies are strings a browser uses to maintain state between HTTP calls. State can be maintained throughout a client session or longer if a cookie expiration date is included. |
| OWA_CUSTOM | The authorize function used by cookies. |
| OWA_IMAGE | Subprograms that obtain the coordinates where a user clicked an image. Use this package when you have an image map whose destination links invoke a PL/SQL gateway. |
| OWA_OPT_LOCK | Subprograms that impose database optimistic locking strategies to prevent lost updates. Lost updates can otherwise occur if a user selects, and then attempts to update, a row whose values have been changed in the meantime by another user. |
| OWA_PATTERN | Subprograms that perform string matching and string manipulation with regular expressions. |
| OWA_SEC | Subprograms used by the PL/SQL gateway for authenticating requests. |
| OWA_TEXT | Subprograms used by package OWA_PATTERN for manipulating strings. You can also use them directly. |
| OWA_UTIL | The following types of utility subprograms:<br>▪ Dynamic SQL utilities to produce pages with dynamically generated SQL code.<br>▪ HTML utilities to retrieve the values of CGI environment variables and perform URL redirects.<br>▪ Date utilities for correct date-handling. Date values are simple strings in HTML, but must be properly treated as an Oracle Database datatype. |
| WPG_DOCLOAD | Subprograms that download documents from a document repository that you define using the DAD configuration. |

**See Also:** *Oracle Database PL/SQL Packages and Types Reference* for syntax, descriptions, and examples for the PL/SQL Web Toolkit packages

## Using the mod_plsql Gateway

As explained in detail in the *Oracle Application Server mod_plsql User's Guide*, mod_plsql maps Web client requests to PL/SQL stored procedures over HTTP. Refer to this documentation for instructions.

**See Also:**

- *Oracle Application Server mod_plsql User's Guide* to learn how to configure and use mod_plsql
- *Oracle HTTP Server Administrator's Guide* to obtain mod_plsql reference material

# Generating HTML Output with PL/SQL

Traditionally, PL/SQL Web applications use function calls to generate each HTML tag for output. These functions are part of the PL/SQL Web Toolkit packages that come with Oracle Database. Example 11–1 illustrates how to generate a simple HTML page by calling the `HTP` functions that correspond to each HTML tag.

***Example 11–1 Displaying HTML Tags with HTP Functions***

```
CREATE OR REPLACE PROCEDURE html_page
IS
BEGIN
  HTP.HTMLOPEN;                              -- generates <HTML>
  HTP.HEADOPEN;                              -- generates <HEAD>
  HTP.TITLE('Title');                        -- generates <TITLE>Hello</TITLE>
  HTP.HEADCLOSE;                             -- generates </HTML>

  -- generates <BODY TEXT="#000000" BGCOLOR="#FFFFFF">
  HTP.BODYOPEN( cattributes => 'TEXT="#000000" BGCOLOR="#FFFFFF"');

  -- generates <H1>Heading in the HTML File</H1>
  HTP.HEADER(1, 'Heading in the HTML File');

  HTP.PARA;                                   -- generates <P>
  HTP.PRINT('Some text in the HTML file.');
  HTP.BODYCLOSE;                             -- generates </BODY>
  HTP.HTMLCLOSE;                             -- generates </HTML>
END;
```

An alternative to making function calls that correspond to each tag is to use the `HTP.PRINT` function to print the text and tags together. Example 11–2 illustrates this technique.

***Example 11–2 Displaying HTML Tags with HTP.PRINT***

```
CREATE OR REPLACE PROCEDURE html_page2
IS
BEGIN
  HTP.PRINT('<html>');
  HTP.PRINT('<head>');
  HTP.PRINT('<meta http-equiv="Content-Type" content="text/html">');
  HTP.PRINT('<title>Title of the HTML File</title>');
  HTP.PRINT('</head>');

  HTP.PRINT('<body TEXT="#000000" BGCOLOR="#FFFFFF">');
  HTP.PRINT('<h1>Heading in the HTML File</h1>');
  HTP.PRINT('<p>Some text in the HTML file.');
  HTP.PRINT('</body>');

  HTP.PRINT('</html>');
END;
```

Chapter 12, "Developing PL/SQL Server Pages" describes an additional method for delivering using PL/SQL to generate HTML content. PL/SQL server pages enables you to build on your knowledge of HTML tags and avoid learning a new set of function calls. In an application written as a set of PL/SQL server pages, you can still use functions from the PL/SQL Web toolkit to do the following:

- Simplify the processing involved in displaying tables
- Store persistent data (cookies)
- Work with CGI protocol internals

## Passing Parameters to a PL/SQL Web Application

To be useful in a wide variety of situations, a Web application must be interactive enough to allow user choices. To keep the attention of impatient Web surfers, you should streamline the interaction so that users can specify these choices very simply, without excessive decision-making or data entry.

The main methods of passing parameters to PL/SQL Web applications are:

- Using HTML form tags. The user fills in a form on one Web page, and all the data and choices are transmitted to a stored procedure when the user clicks the `Submit` button on the page.
- Hard-coded in the URL. The user clicks on a link, and a set of predefined parameters are transmitted to a stored procedure. Typically, you would include separate links on your Web page for all the choices that the user might want.

This section contains the following topics:

- Passing List and Dropdown List Parameters from an HTML Form
- Passing Radio Button and Checkbox Parameters from an HTML Form
- Passing Entry Field Parameters from an HTML Form
- Passing Hidden Parameters from an HTML Form
- Uploading a File from an HTML Form
- Submitting a Completed HTML Form
- Handling Missing Input from an HTML Form
- Maintaining State Information Between Web Pages

### Passing List and Dropdown List Parameters from an HTML Form

List boxes and drop-down lists are implemented with the HTML tag `<SELECT>`.

Use a list box for a large number of choices or to allow multiple selections. List boxes are good for showing items in alphabetical order so that users can find an item quickly without reading all the choices.

Use a drop-down list in the following situations:

- There are a small number of choices
- Screen space is limited.
- Choices are in an unusual order.

The drop-down captures the attention of first-time users and makes them read the items. If you keep the choices and order consistent, then users can memorize the

motion of selecting an item from the drop-down list, allowing them to make selections quickly as they gain experience. Example 11–3 shows a simple drop-down list.

***Example 11–3 HTML Drop-Down List***

```
<form>
<select name="seasons">
<option value="winter">Winter
<option value="spring">Spring
<option value="summer">Summer
<option value="fall">Fall
</select>
```

## Passing Radio Button and Checkbox Parameters from an HTML Form

Radio buttons pass either a null value (if none of the radio buttons in a group is checked), or the value specified on the radio button that is checked.

To specify a default value for a set of radio buttons, you can include the `CHECKED` attribute in one of the `INPUT` tags, or include a `DEFAULT` clause on the parameter within the stored procedure. When setting up a group of radio buttons, be sure to include a choice that indicates "no preference", because once the user selects a radio button, they can still select a different one, but they cannot clear the selection completely. For example, include a "Don't Care" or "Don't Know" selection along with "Yes" and "No" choices, in case someone makes a selection and then realizes it was wrong.

Checkboxes need special handling, because your stored procedure might receive a null value, a single value, or multiple values:

All the checkboxes with the same `NAME` attribute make up a checkbox group. If none of the checkboxes in a group is checked, the stored procedure receives a null value for the corresponding parameter.

If one checkbox in a group is checked, the stored procedure receives a single `VARCHAR2` parameter.

If more than one checkbox in a group is checked, the stored procedure receives a parameter with the PL/SQL type `TABLE OF VARCHAR2`. You must declare a type like this, or use a predefined one like `OWA_UTIL.IDENT_ARR`. To retrieve the values, use a loop:

```
CREATE OR REPLACE PROCEDURE handle_checkboxes ( checkboxes owa_util.ident_arr )
AS
BEGIN
  ...
  FOR i IN 1..checkboxes.count
  LOOP
    htp.print('<p>Checkbox value: ' || checkboxes(i));
  END LOOP;
  ...
END;
/
SHOW ERRORS;
```

## Passing Entry Field Parameters from an HTML Form

Entry fields require the most validation, because a user might enter data in the wrong format, out of range, and so on. If possible, validate the data on the client side using

dynamic HTML or Java, and format it correctly for the user or prompt them to enter it again.

For example:

- You might prevent the user from entering alphabetic characters in a numeric entry field, or from entering characters once a length limit is reached.
- You might silently remove spaces and dashes from a credit card number if the stored procedure expects the value in that format.
- You might inform the user immediately when they type a number that is too large, so that they can retype it.

Because you cannot always rely on such validation to succeed, code the stored procedures to deal with these cases anyway. Rather than forcing the user to use the `Back` button when they enter wrong data, display a single page with an error message and the original form with all the other values filled in.

For sensitive information such as passwords, a special form of the entry field, `<INPUT TYPE=PASSWORD>`, hides the text as it is typed in.

For example, the following procedure accepts two strings as input. The first time it is called, the user sees a simple form prompting for the input values. When the user submits the information, the same procedure is called again to check if the input is correct. If the input is OK, the procedure processes it. If not, the procedure prompts for new input, filling in the original values for the user.

```
-- Store a name and associated zip code in the database.
CREATE OR REPLACE PROCEDURE associate_name_with_zipcode
(
  name VARCHAR2 DEFAULT NULL,
  zip VARCHAR2 DEFAULT NULL
)
AS
  booktitle VARCHAR2(256);
BEGIN
-- Both entry fields must contain a value. The zip code must be 6 characters.
-- (In a real program you would perform more extensive checking.)
  IF name IS NOT NULL AND zip IS NOT NULL AND length(zip) = 6 THEN
    store_name_and_zipcode(name, zip);
    htp.print('<p>The person ' || name || ' has the zip code ' || zip || '.');
-- If the input was OK, we stop here and the user does not see the form again.
    RETURN;
  END IF;

-- If some data was entered, but it is not correct, show the error message.
  IF (name IS NULL AND zip IS NOT NULL)
    OR (name IS NOT NULL AND zip IS NULL)
    OR (zip IS NOT NULL AND length(zip) != 6)
  THEN
    htp.print('<p><b>Please re-enter the data. Fill in all fields, and use a
              6-digit zip code.</b>');
  END IF;

-- If the user has not entered any data, or entered bad data, prompt for
-- input values.

-- Make the form call the same procedure to check the input values.
  htp.formOpen( 'scott.associate_name_with_zipcode', 'GET');
  htp.print('<p>Enter your name:</td>');
  htp.print('<td valign=center><input type=text name=name value="' || name ||
```

```
 '">');
  htp.print('<p>Enter your zip code:</td>');
  htp.print('<td valign=center><input type=text name=zip value="' || zip || '">');
  htp.formSubmit(NULL, 'Submit');
  htp.formClose;
END;
/
SHOW ERRORS;
```

## Passing Hidden Parameters from an HTML Form

One technique for passing information through a sequence of stored procedures, without requiring the user to specify the same choices each time, is to include hidden parameters in the form that calls a stored procedure. The first stored procedure places information, such as a user name, into the HTML form that it generates. The value of the hidden parameter is passed to the next stored procedure, as if the user had entered it through a radio button or entry field.

Other techniques for passing information from one stored procedure to another include:

- Sending a "cookie" containing the persistent information to the browser. The browser then sends this same information back to the server when accessing other Web pages from the same site. Cookies are set and retrieved through the HTTP headers that are transferred between the browser and the Web server before the HTML text of each Web page.
- Storing the information in the database itself, where later stored procedures can retrieve it. This technique involves some extra overhead on the database server, and you must still find a way to keep track of each user as multiple users access the server at the same time.

## Uploading a File from an HTML Form

You can use an HTML form to choose a file on a client system, and transfer it to the server. A stored procedure can insert the file into the database as a `CLOB`, `BLOB`, or other type that can hold large amounts of data.

The PL/SQL Web toolkit and the PL/SQL gateway have the notion of a "document table" that holds uploaded files.

**See Also:** *mod_plsql User's Guide*

## Submitting a Completed HTML Form

By default, an HTML form must have a `Submit` button, which transmits the data from the form to a stored procedure or CGI program. You can label this button with text of your choice, such as "Search", "Register", and so on.

You can have multiple forms on the same page, each with its own form elements and `Submit` button. You can even have forms consisting entirely of hidden parameters, where the user makes no choice other than clicking the button.

Using JavaScript or other scripting languages, you can do away with the Submit button and have the form submitted in response to some other action, such as selecting from a drop-down list. This technique is best when the user only makes a single selection, and the confirmation step of the `Submit` button is not essential.

## Handling Missing Input from an HTML Form

When an HTML form is submitted, your stored procedure receives null parameters for any form elements that are not filled in. For example, null parameters can result from an empty entry field, a set of checkboxes, radio buttons, or list items with none checked, or a `VALUE` parameter of "" (empty quotation marks).

Regardless of any validation you do on the client side, always code stored procedures to handle the possibility that some parameters are null:

- Use a `DEFAULT` clause in all parameter declarations, to prevent an exception when the stored procedure is called with a missing form parameter. You can set the default to zero for numeric values (when that makes sense), and use `DEFAULT NULL` when you want to check whether or not the user actually specifies a value.
- Before using an input parameter value that has a `DEFAULT NULL` declaration, check if it is null.
- Make the procedure generate sensible results even when not all input parameters are specified. You might leave some sections out of a report, or display a text string or image in a report to indicate where parameters were not specified.
- Provide a way to fill in the missing values and run the stored procedure again, directly from the results page. For example, you could include a link that calls the same stored procedure with an additional parameter, or display the original form with its values filled in as part of the output.

## Maintaining State Information Between Web Pages

Web applications are particularly concerned with the idea of **state**, the set of data that is current at a particular moment in time. It is easy to lose state information when switching from one Web page to another, which might result in asking the user to make the same choices over and over.

You can pass state information between dynamic Web pages using HTML forms. The information is passed as a set of name-value pairs, which are turned into stored procedure parameters for you.

If the user has to make multiple selections, or one selection from many choices, or it is important to avoid an accidental selection, use an HTML form. After the user makes and reviews all the choices, they confirm the choices with the `Submit` button. Subsequent pages can use forms with hidden parameters (`<INPUT TYPE=HIDDEN>` tags) to pass these choices from one page to the next.

If the user is only considering one or two choices, or the decision points are scattered throughout the Web page, you can save the user from hunting around for the `Submit` button by representing actions as hyperlinks and including any necessary name-value pairs in the query string (the part following the `?` within a URL).

An alternative way to main state information is to use Oracle Application Server and its `mod_ose` module. This approach lets you store state information in package variables that remain available as a user moves around a Web site.

> **See Also:** The Oracle Application Server documentation set at `http://www.oracle.com/technology/documentation`

# Performing Network Operations within PL/SQL Stored Procedures

While built-in PL/SQL features are focused on traditional database operations and programming logic, Oracle Database provides packages that open up Internet computing to PL/SQL programmers.

This section contains the following topics:


- Sending E-Mail from PL/SQL
- Getting a Host Name or Address from PL/SQL
- Working with TCP/IP Connections from PL/SQL
- Retrieving the Contents of an HTTP URL from PL/SQL
- Working with Tables, Image Maps, Cookies, and CGI Variables from PL/SQL


## Sending E-Mail from PL/SQL

You can send e-mail from a PL/SQL program or stored procedure with the `UTL_SMTP` package. You can read about this package in the *Oracle Database PL/SQL Packages and Types Reference*.

The following code example illustrates how the SMTP package might be used by an application to send e-mail. The application connects to an SMTP server at port 25 and sends a simple text message.

```
PROCEDURE send_test_message
IS
    mailhost    VARCHAR2(64) := 'mailhost.fictional-domain.com';
    sender      VARCHAR2(64) := 'me@fictional-domain.com';
    recipient   VARCHAR2(64) := 'you@fictional-domain.com';
    mail_conn  utl_smtp.connection;
BEGIN
    mail_conn := utl_smtp.open_connection(mailhost, 25);
    utl_smtp.helo(mail_conn, mailhost);
    utl_smtp.mail(mail_conn, sender);
    utl_smtp.rcpt(mail_conn, recipient);
-- If we had the message in a single string, we could collapse
-- open_data(), write_data(), and close_data() into a single call to data().
    utl_smtp.open_data(mail_conn);
    utl_smtp.write_data(mail_conn, 'This is a test message.' || chr(13));
    utl_smtp.write_data(mail_conn, 'This is line 2.' || chr(13));
    utl_smtp.close_data(mail_conn);
    utl_smtp.quit(mail_conn);
    EXCEPTION
        WHEN OTHERS THEN
           -- Insert error-handling code here
           NULL;
END;
```

## Getting a Host Name or Address from PL/SQL

You can determine the host name of the local machine, or the IP address of a given host name from a PL/SQL program or stored procedure using the `UTL_INADDR` package. You can find details about this package in the *Oracle Database PL/SQL Packages and Types Reference*. You use the results in calls to the `UTL_TCP` package.

## Working with TCP/IP Connections from PL/SQL

You can open TCP/IP connections to machines on the network, and read or write to the corresponding sockets, using the UTL_TCP package. You can find details about this package in the *Oracle Database PL/SQL Packages and Types Reference*.

## Retrieving the Contents of an HTTP URL from PL/SQL

You can retrieve the contents of an HTTP URL using the UTL_HTTP package. The contents are typically in the form of HTML-tagged text, but may be plain text, a JPEG image, or any sort of file that is downloadable from a Web server. You can find details about this package in the *Oracle Database PL/SQL Packages and Types Reference*.

The UTL_HTTP package lets you:

- Control the details of the HTTP session, including header lines, cookies, redirects, proxy servers, IDs and passwords for protected sites, and CGI parameters through the GET or POST methods.
- Speed up multiple accesses to the same Web site using HTTP 1.1 persistent connections.
- Construct and interpret URLs for use with UTL_HTTP through the ESCAPE and UNESCAPE functions in the UTL_URL package.

Typically, developers have used Java or Perl to perform these operations; this package lets you do them with PL/SQL.

```
CREATE OR REPLACE PROCEDURE show_url
(
    url      IN VARCHAR2,
    username IN VARCHAR2 DEFAULT NULL,
    password IN VARCHAR2 DEFAULT NULL
) AS
    req       utl_http.req;
    resp      utl_http.resp;
    name      VARCHAR2(256);
    value     VARCHAR2(1024);
    data      VARCHAR2(255);
    my_scheme VARCHAR2(256);
    my_realm  VARCHAR2(256);
    my_proxy  BOOLEAN;
BEGIN
-- When going through a firewall, pass requests through this host.
-- Specify sites inside the firewall that don't need the proxy host.
  utl_http.set_proxy('proxy.my-company.com', 'corp.my-company.com');

-- Ask UTL_HTTP not to raise an exception for 4xx and 5xx status codes,
-- rather than just returning the text of the error page.
  utl_http.set_response_error_check(FALSE);

-- Begin retrieving this Web page.
  req := utl_http.begin_request(url);

-- Identify ourselves. Some sites serve special pages for particular browsers.
  utl_http.set_header(req, 'User-Agent', 'Mozilla/4.0');

-- Specify a user ID and password for pages that require them.
  IF (username IS NOT NULL) THEN
    utl_http.set_authentication(req, username, password);
  END IF;
```

```
  BEGIN
-- Start receiving the HTML text.
    resp := utl_http.get_response(req);

-- Show the status codes and reason phrase of the response.
    dbms_output.put_line('HTTP response status code: ' || resp.status_code);
    dbms_output.put_line('HTTP response reason phrase: ' || resp.reason_phrase);

-- Look for client-side error and report it.
    IF (resp.status_code >= 400) AND (resp.status_code <= 499) THEN

-- Detect whether the page is password protected, and we didn't supply
-- the right authorization.
      IF (resp.status_code = utl_http.HTTP_UNAUTHORIZED) THEN
        utl_http.get_authentication(resp, my_scheme, my_realm, my_proxy);
        IF (my_proxy) THEN
          dbms_output.put_line('Web proxy server is protected.');
          dbms_output.put('Please supply the required ' || my_scheme ||
            ' authentication username/password for realm ' || my_realm ||
            ' for the proxy server.');
        ELSE
          dbms_output.put_line('Web page ' || url || ' is protected.');
          dbms_output.put('Please supplied the required ' || my_scheme ||
            ' authentication username/password for realm ' || my_realm ||
            ' for the Web page.');
        END IF;
      ELSE
        dbms_output.put_line('Check the URL.');
      END IF;

      utl_http.end_response(resp);
      RETURN;

-- Look for server-side error and report it.
    ELSIF (resp.status_code >= 500) AND (resp.status_code <= 599) THEN

      dbms_output.put_line('Check if the Web site is up.');
      utl_http.end_response(resp);
      RETURN;

    END IF;

-- The HTTP header lines contain information about cookies, character sets,
-- and other data that client and server can use to customize each session.
    FOR i IN 1..utl_http.get_header_count(resp) LOOP
      utl_http.get_header(resp, i, name, value);
      dbms_output.put_line(name || ': ' || value);
    END LOOP;

-- Keep reading lines until no more are left and an exception is raised.
    LOOP
      utl_http.read_line(resp, value);
      dbms_output.put_line(value);
    END LOOP;
  EXCEPTION
    WHEN utl_http.end_of_body THEN
    utl_http.end_response(resp);
  END;

END;
```

```
/
SET serveroutput ON
-- The following URLs illustrate the use of this procedure,
-- but these pages do not actually exist. To test, substitute
-- URLs from your own Web server.
exec show_url('http://www.oracle.com/no-such-page.html')
exec show_url('http://www.oracle.com/protected-page.html')
exec show_url('http://www.oracle.com/protected-page.html', 'scott', 'tiger')
```

## Working with Tables, Image Maps, Cookies, and CGI Variables from PL/SQL

Packages for all of these functions are supplied with Oracle8*i* and higher. You use these packages in combination with the mod_plsql plug-in of Oracle HTTP Server (OHS). You can format the results of a query in an HTML table, produce an image map, set and get HTTP cookies, check the values of CGI variables, and combine other typical Web operations with a PL/SQL program.

Documentation for these packages is not part of the database documentation library. The location of the documentation depends on the particular application server you are running. To get started with these packages, look at their procedure names and parameters using the SQL*Plus `DESCRIBE` command:

```
DESCRIBE HTP;
DESCRIBE HTF;
DESCRIBE OWA_UTIL;
```

# 12

# Developing PL/SQL Server Pages

This section contains the following topics:


- PL/SQL Server Pages: Overview
- Writing a PL/SQL Server Page
- Loading a PL/SQL Server Page into the Database
- Executing a PL/SQL Server Page Through a URL
- Examples of PL/SQL Server Pages
- Debugging PL/SQL Server Page Problems
- Putting PL/SQL Server Pages into Production


## PL/SQL Server Pages: Overview

This section contains the following topics:


- What Are PL/SQL Server Pages and Why Use Them?
- Prerequisites for Developing and Deploying PL/SQL Server Pages
- PSP and the HTP Package
- PSP and Other Scripting Solutions


### What Are PL/SQL Server Pages and Why Use Them?

*PL/SQL Server Pages* (PSP) are server-side scripts that include dynamic content, including the results of SQL queries, inside Web pages. You can author the Web pages in an HTML authoring tool and insert blocks of PL/SQL code.

Example 12–1 shows a simple PL/SQL server page called `simple.psp`.

***Example 12–1 simple.psp***

```
<%@ page language="PL/SQL" %>
<%@ page contentType="text/html" %>
<%@ plsql procedure="show_employees" %>
<%-- This example displays the last name and first name of every
     employee in the hr.employees table. --%>
<%!
  CURSOR emp_cursor IS
  SELECT last_name, first_name
  FROM hr.employees
  ORDER BY last_name;
%>
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html">
<title>List of Employees</title>
</head>
<body TEXT="#000000" BGCOLOR="#FFFFFF">
<h1>List of Employees</h1>
<table width="40%" border="1">
<tr>
<th align="left">Last Name</th>
<th align="left">First Name</th>
</tr>
<%  FOR emp_record IN emp_cursor LOOP %>
  <tr>
  <td> <%= emp_record.last_name %> </td>
  <td> <%= emp_record.first_name %> </td>
  </tr>
<%  END LOOP; %>
</table>
</body>
</html>
```

You can compile and load this script into an Oracle database with the `loadpsp` command-line utility. The following example loads this server page into the `hr` schema, replacing the `show_employees` procedure if it already exists:

```
loadpsp -replace -user hr/hr simple.psp
```

Browser users can execute the `show_employees` procedure through a URL. An HTML page that displays the last and first names of employees in the `hr.employees` table is returned to the browser through the PL/SQL gateway.

Deploying content through PL/SQL Server Pages has the following advantages:

- For developers familiar with PL/SQL, the server pages are the easiest way to create professional Web pages that included database-generated content. You can develop Web pages as normal and then embed PL/SQL code in the HTML.
- PSP can be more convenient than using the `HTP` and `HTF` packages to write out HTML content line by line.
- Because processing is performed on the database server, the client browser receives a plain HTML page with no special script tags. You can support all browsers and browser levels equally.
- Network traffic is efficient because use of PSP minimizes the number of database round-trips.
- You can write content quickly and follow a rapid, iterative development process. You maintain central control of the software, with only a Web browser required on the client machine.

## Prerequisites for Developing and Deploying PL/SQL Server Pages

To develop and deploy PL/SQL server pages, you must meet the following prerequisites:

- To write a PL/SQL server page you need access to a text editor or HTML authoring tool for writing the script. No other development tool is required.
- To load a PL/SQL server page you need:

  - An account on an Oracle database in which to load the server pages.
  - Execution rights to the `loadpsp` command-line utility, which is located in `$ORACLE_HOME/bin`.
- To deploy the server pages you must use mod_plsql. As explained in "PL/SQL Web Toolkit" on page 11-2, the gateway makes use of the PL/SQL Web Toolkit.

  **See Also:**

  - "Using the mod_plsql Gateway" on page 11-3

## PSP and the HTP Package

You can enable browser users to execute PL/SQL program units through HTTP in the following ways:

- By writing an HTML page with embedded PL/SQL code and compiling it as a PL/SQL server page. You may call procedures from the PL/SQL Web Toolkit, but not to generate every line of HTML output.
- By writing a complete stored procedure that produces HTML by calling the `HTP` and `OWA_*` packages in the PL/SQL Web Toolkit. This technique is described in "Generating HTML Output with PL/SQL" on page 11-4.

Thus, you must choose which technique to use when writing your Web application. The key factors in choosing between these techniques are:

- What source are you using as a starting point?
  - If you have a large body of HTML, and if you want to include dynamic content or make it the front end of a database application, then use PSP.
  - If you have a large body of PL/SQL code that produces formatted output, then you may find it more convenient to produce HTML tags by changing your print statements to call the `HTP` package of the PL/SQL Web Toolkit.
- What is the fastest and most convenient authoring environment for your group?
  - If most work is done using HTML authoring tools, then use PSP.
  - If you use authoring tools that produce PL/SQL code, then it might be less convenient to use PSP.

## PSP and Other Scripting Solutions

Scripting solutions can be client-side or server-side. JavaScript is one of the most popular client-side scripting language. PSP fully supports JavaScript. Because any kind of tags can be passed unchanged to the browser through a PL/SQL server page, you can include JavaScript or other client-side script code in a PL/SQL server page.

Java Server Pages (JSP) and Active Server Pages (ASP) are two of the most popular server-side scripting solutions. Note the following points of comparison with PSP:

- Java server pages are loosely analogous to PSP pages; Java servlets are analogous to PL/SQL packages. PSP uses the same script tag syntax as JSP to make it easy to switch back and forth.
- PSP uses syntax that is similar to ASP, although not identical. Typically, you must translate from VBScript or JScript to PL/SQL. The best candidates for migration are pages that use the Active Data Object (ADO) interface to perform database operations.

**Note:** You cannot mix PL/SQL server pages with other server-side script features, such as server-side includes. In many cases, you can get the same results by using the corresponding PSP features.

## Writing a PL/SQL Server Page

To write a PL/SQL server page, you can start with an existing Web page or with an existing stored procedure. Either way, with a few additions and changes you can create dynamic Web pages that perform database operations and display the results.

The file for a PL/SQL server page must have the extension `.psp`. It can contain whatever content you choose, with text and tags interspersed with PSP directives, declarations, and scriptlets. A server page can take the following forms:

- In the simplest case, it is an HTML file. Compiling it as a PL/SQL server page produces a stored procedure that outputs exactly the same HTML file.
- In the most complex case, it is a PL/SQL procedure that generates all the content of the Web page, including the tags for title, body, and headings.
- In the typical case, it is a mixture of HTML (providing the static parts of the page) and PL/SQL (providing the dynamic content).

The order and placement of the PSP directives and declarations is usually not significant. It becomes significant only when another file is included. For ease of maintenance, it is recommended that you place the directives and declarations together near the beginning of the file.

Table 12–1 lists the PSP elements and directs you to the section that discusses how to use them. The section "Quoting and Escaping Strings in a PSP Script" on page 12-11 describes how to quote strings that are used in various PSP elements.

***Table 12–1 PSP Elements***

| PSP Element | Name | Specifies . . . | Section |
|---|---|---|---|
| `<%@ page ... %>` | Page Directive | Characteristics of the PL/SQL server page. | "Specifying Basic Server Page Characteristics" on page 12-5 |
| `<%@ parameter ... %>` | Parameter Directive | The name, and optionally the type and default, for each parameter expected by the PSP stored procedure. | "Accepting User Input" on page 12-7 |
| `<%@ plsql ... %>` | Procedure Directive | The name of the stored procedure produced by the PSP file. | "Naming the PL/SQL Stored Procedure" on page 12-8 |
| `<%@ include ... %>` | Include Directive | The name of a file to be included at a specific point in the PSP file. | "Including the Contents of Other Files" on page 12-8 |
| `<%! ... %>` | Declaration Block | The declaration for a set of PL/SQL variables that are visible throughout the page, not just within the next `BEGIN/END` block. | "Declaring Global Variables in a PSP Script" on page 12-9 |
| `<% ... %>` | Code Block | A set of PL/SQL statements to be executed when the procedure is run. | "Specifying Executable Statements in a PSP Script" on page 12-9 |
| `<%= ... %>` | Expression Block | A single PL/SQL expression, such as a string, arithmetic expression, function call, or combination of these. | "Substituting an Expression Result in a PSP Script" on page 12-10 |
| `<%-- ... --%>` | Comment | A comment in a PSP script. | "Including Comments in a PSP Script" on page 12-11 |

**Note:** If you are familiar with dynamic HTML and want to start coding right away, you can jump forward to"Examples of PL/SQL Server Pages" on page 12-15.

## Specifying Basic Server Page Characteristics

Use the `<%@ page ... %>` directive to specify characteristics of the PL/SQL server page such as the following:

- What scripting language it uses.
- What type of information (MIME type) it produces.
- What code to run to handle all uncaught exceptions. This might be an HTML file with a friendly message, renamed to a `.psp` file. You must specify this same file name in the `loadpsp` command that compiles the main PSP file. You must specify exactly the same name in both the `errorPage` directive and in the `loadpsp` command, including any relative path name such as `../include/`.

The following code shows the syntax of the `page` directive (note that the attribute names `contentType` and `errorPage` are case-sensitive):

```
<%@ page [language="PL/SQL"] [contentType="content type string"] charset="encoding" [errorPage="file.psp"] %>
```

### Specifying the Scripting Language

To identify a file as a PL/SQL server page, include the following directive somewhere in the file:

```
<%@ page language="PL/SQL" %>
```

This directive is for compatibility with other scripting environments. Example 12–1 shows an example of a simple PL/SQL server page that includes the language directive.

### Returning Data to the Client

You have the following basic options when specifying the type of data to return to the client browser:

- Returning HTML
- Returning XML, Text, or Other Document Types
- Returning Pages Containing Different Character Sets

**Returning HTML** The PL/SQL parts of a PL/SQL server page are enclosed within special delimiters. All other content is passed along verbatim—including any whitespace—to the browser. To display text or HTML tags, write it as you would a typical Web page. You do not need to call any output functions. As illustration, the server page in Example 12–1 returns the HTML page shown in Example 12–2, except that it includes the table rows for the queried employees.

***Example 12–2 Sample Returned HTML Page***

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html">
<title>List of Employees</title>
</head>
<body TEXT="#000000" BGCOLOR="#FFFFFF">
```

```
<h1>List of Employees</h1>
<table width="40%" border="1">
<tr>
<th align="left">Last Name</th>
<th align="left">First Name</th>
</tr>

  <!-- result set of query of hr.employees inserted here -->

</table>
</body>
</html>
```

Sometimes you might want to display one line of output or another, or change the value of an attribute, based on a condition. You can include control structures and variable substitution inside the PSP delimiters, as shown in the following code fragment from Example 12–1:

```
<%  FOR emp_record IN emp_cursor LOOP %>
  <tr>
  <td> <%= emp_record.last_name %> </td>
  <td> <%= emp_record.first_name %> </td>
  </tr>
<%  END LOOP; %>
```

**Returning XML, Text, or Other Document Types** By default, the PL/SQL gateway transmits files as HTML documents so that the browser interprets the HTML tags. If you want the browser to interpret the document as XML, plain text (with no formatting), or some other document type, then include the following directive:

```
<%@ page contentType="MIMEtype" %>
```

The attribute name is case-sensitive, so be sure to capitalize it as `contentType`. Insert `text/html`, `text/xml`, `text/plain`, `image/jpeg`, or some other MIME type that the browser or other client program recognizes. Users may have to configure their browsers to recognize some MIME types. The following shows an example of a directive for an Excel spreadsheet:

```
<%@ page contentType="application/vnd.ms-excel" %>
```

Typically, a PL/SQL server page is intended to be displayed in a Web browser. It can also be retrieved and interpreted by a program that can make HTTP requests, such as a a Java or Perl client.

**Returning Pages Containing Different Character Sets** By default, the PL/SQL gateway transmits files with the character set defined by the PL/SQL gateway. To convert the data to a different character set for browser display, include the following directive:

```
<%@ page charset="encoding" %>
```

Specify `Shift_JIS`, `Big5`, `UTF-8`, or another encoding that the client program recognizes.

You must also configure the character set setting in the database accessor descriptor (DAD) of the PL/SQL gateway. Users may have to select the same encoding in their browsers to see the data displayed properly. For example, a database in Japan might have a database character set that uses the `EUC` encoding, but the Web browsers are configured to display `Shift_JIS` encoding.

### Handling Script Errors

When writing PL/SQL server pages, be mindful of the following types of errors:

- HTML syntax errors. Any errors in HTML markup are handled by the browser. The `loadpsp` utility does not check for them.
- PL/SQL syntax errors. If you make a syntax error in the PL/SQL code, the `loadpsp` utility stops and displays the line number, column number, and a brief message. You must fix the error before continuing. Note that any previous version of the stored procedure can be erased when you attempt to replace it with a script that contains a syntax error. You might want to use one database for prototyping and debugging, then load the final stored procedure into a different database for production. You can switch databases using a command-line flag without changing any source code.
- Runtime errors. To handle database errors that occur when the script runs, you can include PL/SQL exception-handling code within a PSP file and have any unhandled exceptions bring up a special PL/SQL server page. Use the `errorPage` attribute (note that the name is case-sensitive) of the `<%@ page ... %>` directive to specify the page name.

  The page for unhandled exceptions is a PL/SQL server page with extension `.psp`. The error procedure does not receive any parameters, so to determine the cause of the error, it can call the `SQLCODE` and `SQLERRM` functions. You can also display a standard HTML page without any scripting when an error occurs, but you must still give it the extension `.psp` and load it into the database as a stored procedure.

The following example shows a directive that specifies `errors.psp` as the page to run when errors are encountered:

```
<%@ page language="PL/SQL" contentType="text/html" errorPage="errors.psp" %>
```

## Accepting User Input

To set up parameter passing for a PL/SQL server page, include a directive with the following syntax:

```
<%@ plsql parameter="parameter name" [type="PL/SQL type"] [default="value"] %>
```

Example 12–9 shows an example of a script that includes the parameter directive.

By default, parameters are of type `VARCHAR2`. To use a different type, include a `type="PL/SQL type"` attribute within the directive, as in the following example:

```
<%@ plsql parameter="p_employee_id" type="NUMBER" %>
```

To set a default value, so that the parameter becomes optional, include a `default="expression"` attribute in the directive. The values for this attribute are substituted directly into a PL/SQL statement, so any strings must be single-quoted, and you can use special values such as `null`, as in the following example:

```
<%@ plsql parameter="p_last_name" default="null" %>
```

User input comes encoded in the URL that retrieves the HTML page. You can generate the URL by hard-coding it in an HTML link, or by calling your page as the action of an HTML form. Your page receives the input as parameters to a PL/SQL stored procedure. For example, assume that you change the first few lines of Example 12–1 to include a parameter directive as follows, and then load it into the database:

```
<%@ page language="PL/SQL" %>
<%@ page contentType="text/html" %>
```

```
<%@ plsql parameter="p_employee_id" default="null" type="NUMBER" %>
<%@ plsql procedure="show_employees" %>
<%!
  CURSOR emp_cursor IS
  SELECT last_name, first_name
  FROM hr.employees
  WHERE employee_id = p_employee_id
  ORDER BY last_name;
%>
```

If the PL/SQL gateway were configured so that you could execute procedures by calling `http://www.host.com/pls/proc_name`, where `proc_name` is the name of a procedure, then you could pass `200` for parameter `p_employee_id` as follows:

```
http://www.host.com/pls/show_employees?p_employee_id=200
```

## Naming the PL/SQL Stored Procedure

Each top-level PL/SQL server page corresponds to a stored procedure within the server. When you load the page with `loadpsp`, the utility creates a PL/SQL stored procedure. By default, the procedure is given the same name as the PSP script, except with the `.psp` extension removed. Thus, if your script is named `hello_world.psp`, then by default the utility creates a procedure named `hello_world`.

To give the procedure a name that is different from the script name, include the following directive, where *`procname`* is the name of a procedure:

```
<%@ plsql procedure="procname" %>
```

Example 12–1 includes the following directive, which gives the stored procedure the name `show_employees`:

```
<%@ plsql procedure="show_employees" %>
```

Thus, you could name the file `empnames.psp` or anything else that ends with `*.psp`, but the procedure is created as `show_employees`. Note that it is the name of the *procedure*, not the name of the PSP script, that you include in the URL.

## Including the Contents of Other Files

You can set up an include mechanism to pull in the contents of other files, typically containing either static HTML content or more PL/SQL scripting code. Insert the following directive at the point where the content of the other file should appear, replacing *`filename`* with the name of the file to be included:

```
<%@ include file="filename" %>
```

The included file must have an extension other than `.psp`. You must specify exactly the same name in both the `include` directive and in the `loadpsp` command, including any relative path name such as `../include/`.

Because the files are processed when you load the stored procedure into the database, the substitution is performed only once, not whenever the page is served. Therefore, changes to the included files that occur after the page is loaded into the database are not displayed when the procedure is executed.

You can use the include feature to pull in libraries of code, such as a navigation banners, footers, tables of contents, and so forth into multiple files. Alternatively, you can use this feature as a macro capability to include the same section of script code in more than one place in a page. The following example includes an HTML footer:

```
<%@ include file="footer.htm" %>
```

Note the following characteristics of included files:

- You can use any names and extensions for the included files. For example, you could include a file called `products.txt`.
- If the included files contain PL/SQL scripting code, then they do not need their own set of directives to identify the procedure name, character set, and so on.
- When specifying the names of files to the `loadpsp` utility, you must include the names of all included files also. Specify the names of included files before the names of any `.psp` files.

## Declaring Global Variables in a PSP Script

You can use the `<%! ... %>` directive to define a set of PL/SQL variables that are visible throughout the page, not just within the next `BEGIN/END` block. This element typically spans multiple lines, with individual PL/SQL variable declarations ended by semicolons. The syntax for this directive is as follows:

```
<%! PL/SQL declaration;
    [ PL/SQL declaration; ] ... %>
```

The usual PL/SQL syntax is allowed within the block. The delimiters server as shorthand, enabling you to omit the `DECLARE` keyword. All declarations are available to the code later in the file. Example 12–1 includes the following cursor declaration:

```
<%!
  CURSOR emp_cursor IS
  SELECT last_name, first_name
  FROM hr.employees
  ORDER BY last_name;
%>
```

You can specify multiple declaration blocks; internally, they are all merged into a single block when the PSP file is created as a stored procedure.

You can also use explicit `DECLARE` blocks within the `<% ... %>` delimiters that are explained in "Specifying Executable Statements in a PSP Script" on page 12-9. These declarations are only visible to the following `BEGIN/END` block.

> **Note:** To make things easier to maintain, keep all your directives and declarations together near the beginning of a PL/SQL server page.

## Specifying Executable Statements in a PSP Script

You can use the `<% ... %>` code block directive to execute a set of PL/SQL statements when the stored procedure is run. The following code shows the syntax for executable statements:

```
<% PL/SQL statement;
   [ PL/SQL statement; ] ... %>
```

This element typically spans multiple lines, with individual PL/SQL statements ended by semicolons. The statements can include complete blocks, as in the following example, which calls the `OWA_UTIL.TABLEPRINT` procedure:

```
<% OWA_UTIL.TABLEPRINT(CTABLE => 'hr.employees', CATTRIBUTES => 'border=2',
   CCOLUMNS => 'last_name,first_name', CCLAUSES => 'WHERE employee_id > 100'); %>
```

The statements can also be the bracketing parts of IF/THEN/ELSE or BEGIN/END blocks. When a code block is split into multiple directives, you can put HTML or other directives in the middle, and the middle pieces are conditionally executed when the stored procedure is run. The following code from Example 12–10 provides an illustration of this technique:

```
<% FOR ITEM IN (SELECT product_name, list_price, catalog_url
                FROM product_information
                WHERE list_price IS NOT NULL
                ORDER BY list_price DESC) LOOP
   IF item.list_price > p_minprice THEN
      v_color := '#CCCCFF';
   ELSE
      v_color := '#CCCCCC';
   END IF;
%>
<TR BGCOLOR="<%= v_color %>">
  <TD><A HREF="<%= item.catalog_url %>"><%= item.product_name %></A></TD>
  <TD><BIG><%= item.list_price %></BIG></TD>
</TR>
<% END LOOP; %>
```

All the usual PL/SQL syntax is allowed within the block. The delimiters server as shorthand, letting you omit the DECLARE keyword. All the declarations are available to the code later on in the file.

**Note:** To share procedures, constants, and types across different PL/SQL server pages, compile them into a package in the database by using a plain PL/SQL source file. Although you can reference packaged procedures, constants, and types from PSP scripts, the PSP scripts can only produce standalone procedures, not packages.

## Substituting an Expression Result in a PSP Script

An expression directive outputs a single PL/SQL expression, such as a string, arithmetic expression, function call, or combination of these things. The result is substituted as a string at that spot in the HTML page that is produced by the stored procedure. The expression result must be a string value or be able to be cast to a string. For any types that cannot be implicitly cast, such as DATE, pass the value to the PL/SQL TO_CHAR function.

The syntax of an expression directive is as follows, where the *expression* placeholder is replaced by the desired expression:

```
<%= expression %>
```

Note that you do not need to end the PL/SQL expression with a semicolon.

Example 12–1 includes a directive to print the value of a variable in a row of a cursor:

```
<%= emp_record.last_name %>
```

Compare the preceding example to the equivalent htp.print call in the following example (note especially the semicolon that ends the statement):

```
<% HTP.PRN (emp_record.last_name); %>
```

The content within the <%= ... %> delimiters is processed by the HTP.PRN function, which trims leading or trailing whitespace and requires that you quote literal strings.

Note that you can use concatenation by using the twin pipe symbol (||) as you would in PL/SQL. The following directive shows an example of concatenation:

```
<%= 'The employee last name is ' || emp_record.last_name %>
```

## Quoting and Escaping Strings in a PSP Script

PSP attributes use double quotes to delimit data. When values specified in PSP attributes are used for PL/SQL operations, they are passed exactly as you specify them in the PSP file. Thus, if PL/SQL requires a single-quoted string, then you must specify the string with the single quotes around it—and surround the whole thing with double quotes.

For example, your PL/SQL procedure may use the string Babe Ruth as the default value for a variable. Because the string will be used in PL/SQL, you must enclose it in single quotes as 'Babe Ruth'. If you specify this single-quoted string in the default attribute of a PSP directive, then you must surround it in double quotes as in the following example:

```
<%@ plsql parameter="in_players" default="'Babe Ruth'" %>
```

You can also nest single-quoted strings inside single quotes. In this case, you must *escape* the nested single quotes by specifying the sequence \'. For example:

```
<%@ plsql parameter="in_players" default="'Walter \'Big Train\' Johnson'" %>
```

You can include most characters and character sequences in a PSP file without having them changed by the PSP loader. To include the sequence %>, specify the escape sequence %\>. To include the sequence <%, specify the escape sequence <\%. For example:

```
<%= 'The %\> sequence is used in scripting language: ' || lang_name %>
<%= 'The <\% sequence is used in scripting language: ' || lang_name %>
```

## Including Comments in a PSP Script

To put a comment in the HTML portion of a PL/SQL server page for the benefit of those reading the PSP source code, use the following syntax:

```
<%-- PSP comment text --%>
```

Comments in the preceding form do not appear in the HTML output from the PSP and also do not appear when you query the PL/SQL source code in USER_OBJECTS.

To create a comment that is visible in the HTML output and in the USER_OBJECTS source, place the comment in the HTML and use the normal HTML comment syntax:

```
<!-- HTML comment text -->
```

To include a comment inside a PL/SQL block within a PSP, and to make the comment invisible in the HTML output but visible in USER_OBJECTS, use the normal PL/SQL comment syntax, as in the following example:

```
-- Comment in PL/SQL code
```

Example 12–3 shows a fragment of a PSP file with the three types of comments.

***Example 12–3 Sample Comments in a PSP File***

```
<p>Today we introduce our new model XP-10.
<%--
  This is the project with code name "Secret Project".
  Users viewing the HTML page will not see this PSP script comment.
  The comment is not visible in the USER_OBJECTS source code.
--%>
<!--
  Some pictures of the XP-10.
  Users viewing the HTML page source will see this comment.
  The comment is also visible in the USER_OBJECTS source code.
-->
<%
FOR image_file IN (SELECT pathname, width, height, description
                   FROM image_library WHERE model_num = 'XP-10')
-- Comments interspersed with PL/SQL statements.
-- Users viewing the HTML page source will not see these PL/SQL comments.
-- These comments are visible in the USER_OBJECTS source code.
LOOP
%>
<img src="<%= image_file.pathname %>" width=<% image_file.width %>
height=<% image_file.height %> alt="<% image_file.description %>">
<br>
<% END LOOP; %>
```

# Loading a PL/SQL Server Page into the Database

Use the `loadpsp` utility, which is located in `$ORACLE_HOME/bin`, to load one or more PSP files into the database as stored procedures. Each `.psp` file corresponds to one stored procedure. The pages are compiled and loaded in one step, to speed up the development cycle. The syntax of the `loadpsp` utility as follows:

```
loadpsp [ -replace ] -user username/password[@connect_string]
    [ include_file_name ... ] [ error_file_name ] psp_file_name ...
```

To create procedures with `CREATE OR REPLACE` syntax, use the `-replace` flag.

When you load a PSP file, the loader performs the following actions:

1. Logs on to the database with the specified user name, password, and net service name
2. Creates the stored procedures in the user schema

Include the names of all the include files before the names of the PL/SQL server pages. Also include the name of the file specified in the `errorPage` attribute of the `page` directive. These filenames on the loadpsp command line must match exactly the names specified within the PSP `include` and `page` directives, including any relative path name such as `../include/`. Example 12–4 shows a sample PSP load command.

***Example 12–4 Loading a PSP***

```
loadpsp -replace -user hr/hr@orcl banner.inc error.psp display_order.psp
```

Note the following characteristics of Example 12–4:

- The stored procedure is created in the database `orcl`. The database is accessed as user `hr` with password `hr`, both to create the stored procedure and when the stored procedure is executed.

- `banner.inc` is a file containing boilerplate text and script code that is included by the `.psp` file. The inclusion occurs when the PSP is loaded into the database, not when the stored procedure is executed.
- `error.psp` is a file containing code, text, or both that is processed when an unhandled exception occurs, to present a friendly page rather than an internal error message.
- `display_order.psp` contains the main code and text for the Web page. By default, the corresponding stored procedure is named `display_order`.

## Querying PSP Source Code

After you have loaded a PSP file, you can view the source code in the `USER_SOURCE` or `DBA_SOURCE` tables in the data dictionary. For example, suppose that you load the script in Example 12–1 with the following command:

```
loadpsp -replace -user hr/hr simple.psp
```

If you log on to the database as user `hr`, then you can execute the following query in SQL*Plus to view the source code of the PSP:

```
SET HEADING OFF

SELECT TEXT
FROM   USER_SOURCE
WHERE  NAME = 'SHOW_EMPLOYEES'
ORDER BY LINE;
```

Sample output is shown in Example 12–5. Note that the code generated by `loadpsp` is different from the code in the source file. The `loadpsp` utility has added extra code, mainly calls to the `HTP` package, to the PSP code. The `HTP` package generates the HTML tags for the web page.

***Example 12–5 Output from Query of USER_SOURCE***

```
PROCEDURE show_employees  AS

  CURSOR emp_cursor IS
  SELECT last_name, first_name
  FROM hr.employees
  ORDER BY last_name;

 BEGIN NULL;
owa_util.mime_header('text/html'); htp.prn('
');
htp.prn('
');
htp.prn('

');
htp.prn('
<html>
<head>
<meta http-equiv="Content-Type" content="text/html">
<title>List of Employees</title>
</head>
<body TEXT="#000000" BGCOLOR="#FFFFFF">
<h1>List of Employees</h1>
<table width="40%" border="1">
```

```
<tr>
<th align="left">Last Name</th>
<th align="left">First Name</th>

</tr>
');
  FOR emp_record IN emp_cursor LOOP
htp.prn('
  <tr>
  <td> ');
htp.prn( emp_record.last_name );
htp.prn(' </td>
  <td> ');
htp.prn( emp_record.first_name );
htp.prn(' </td>
  </tr>
');
  END LOOP;
htp.prn('
</table>
</body>
</html>
');
 END;
```

## Executing a PL/SQL Server Page Through a URL

After the PL/SQL server page has been turned into a stored procedure, you can run the procedure by retrieving an HTTP URL through a Web browser or other Internet-aware client program. The virtual path in the URL depends on the way the PL/SQL gateway is configured.

The parameters to the stored procedure are passed through either the `POST` method or the `GET` method of the HTTP protocol. With the `POST` method, the parameters are passed directly from an HTML form and are not visible in the URL. With the `GET` method, the parameters are passed as name-value pairs in the query string of the URL, separated by `&` characters, with most non-alphanumeric characters in encoded format (such as `%20` for a space). You can use the `GET` method to call a PSP page from an HTML form, or you can use a hard-coded HTML link to call the stored procedure with a given set of parameters.

Using `METHOD=GET`, the syntax of the URL looks something like the following:

`http://`*`sitename`*`/`*`schemaname`*`/`*`procname`*`?`*`parmname1`*`=`*`value1`*`&`*`parmname2`*`=`*`value2`*

For example, the following URL includes a `p_lname` and `p_fname` parameter:

```
http://www.host.com/pls/show_employees?p_lname=Ashdown&p_fname=Lance
```

Using `METHOD=POST`, the syntax of the URL does not show the parameters:

`http://`*`sitename`*`/`*`schemaname`*`/`*`procname`*

For example, the following URL specifies a procedure name but does not pass parameters:

```
http://www.host.com/pls/show_employees
```

The `METHOD=GET` format is more convenient for debugging and allows visitors to pass exactly the same parameters when they return to the page through a bookmark.

The METHOD=POST format allows a larger volume of parameter data, and is suitable for passing sensitive information that should not be displayed in the URL. (URLs linger on in the browser's history list and in the HTTP headers that are passed to the next-visited page.) It is not practical to bookmark pages that are called this way.

# Examples of PL/SQL Server Pages

This section shows how you might start with a very simple PL/SQL server page, and produce progressively more complicated versions as you gain more confidence.

As you go through each step, you can follow the instructions in "Loading a PL/SQL Server Page into the Database" on page 12-12 and "Executing a PL/SQL Server Page Through a URL" on page 12-14 to test the examples.

This section contains the following topics:

- Setup for PL/SQL Server Pages Examples
- Printing the Sample Table with a Loop
- Allowing a User Selection
- Using an HTML Form to Call a PL/SQL Server Page

## Setup for PL/SQL Server Pages Examples

These examples use the product_information table in the oe schema, which is described as follows:

```
Table PRODUCT_INFORMATION
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PRODUCT_ID                                NOT NULL NUMBER(6)
 PRODUCT_NAME                                       VARCHAR2(50)
 PRODUCT_DESCRIPTION                                VARCHAR2(2000)
 CATEGORY_ID                                        NUMBER(2)
 WEIGHT_CLASS                                       NUMBER(1)
 WARRANTY_PERIOD                                    INTERVAL YEAR(2) TO MONTH
 SUPPLIER_ID                                        NUMBER(6)
 PRODUCT_STATUS                                     VARCHAR2(20)
 LIST_PRICE                                         NUMBER(8,2)
 MIN_PRICE                                          NUMBER(8,2)
 CATALOG_URL                                        VARCHAR2(50)
```

The examples assume the following:

- Youn have set up mod_plsql as described in "Using the mod_plsql Gateway" on page 11-3.
- You have created a DAD for static authentication of the oe user.
- You can access PL/SQL stored procedures created in the oe schema through the following URL, where proc_name is the name of a stored procedure:
  http://www.host.com/pls/proc_name

For debugging purposes, you can display the complete contents of an SQL table. You can do this with a single call to OWA_UTIL.TABLEPRINT as illustrated in Example 12–6. In subsequent iterations, we use other techniques to gain more control over the presentation.

***Example 12–6 show_prod_simple.psp***

```
<%@ plsql procedure="show_prod_simple" %>
<HTML>
<HEAD><TITLE>Show Contents of product_information (Complete Dump)</TITLE></HEAD>
<BODY>
<%
DECLARE
  dummy BOOLEAN;
BEGIN
  dummy := OWA_UTIL.TABLEPRINT('oe.product_information','border');
END;
%>
</BODY>
</HTML>
```

Load the PSP in Example 12–6 at the command line as follows:

```
loadpsp -replace -user oe/oe show_prod_simple.psp
```

Access the PSP through the following URL:

```
http://www.host.com/pls/show_prod_simple
```

## Printing the Sample Table with a Loop

Example 12–6 loops through the items in the `product_information` table and adjusts the `SELECT` statement to retrieve only a subset of the rows or columns. In this example, we pick a very simple presentation, a set of list items, to avoid any problems from mismatched or unclosed table tags.

***Example 12–7 show_catalog_raw.psp***

```
<%@ plsql procedure="show_prod_raw" %>
<HTML>
<HEAD><TITLE>Show Products (Raw Form)</TITLE></HEAD>
<BODY>
<UL>
<% FOR item IN (SELECT product_name, list_price, catalog_url
                FROM product_information
                WHERE list_price IS NOT NULL
                ORDER BY list_price DESC) LOOP %>
<LI>
Item = <%= item.product_name %><BR>
Price = <%= item.list_price %><BR>
URL = <%= item.catalog_url %><BR>
<% END LOOP; %>
</UL>
</BODY>
</HTML>
```

Example 12–8 shows a more sophisticated variation of Example 12–7 in which formatting is added to the HTML to improve the presentation.

***Example 12–8 show_catalog_pretty.psp***

```
<%@ plsql procedure="show_prod_pretty" %>
<HTML>
<HEAD><TITLE>Show Products (Better Form)</TITLE></HEAD>
<BODY>
<UL>
```

```
<% FOR item IN (SELECT product_name, list_price, catalog_url
                FROM product_information
                WHERE list_price IS NOT NULL
                ORDER BY list_price DESC) LOOP %>
<LI>
Item = <A HREF=<%= item.catalog_url %>><%= item.product_name %></A><BR>
Price = <BIG><%= item.list_price %></BIG><BR>
<% END LOOP; %>
</UL>
</BODY>
</HTML>
```

## Allowing a User Selection

In the previous examples, the HTML page remains the same unless the `product_information` table is updated. Example 12–9 livens up the page by:

- Making it accept a minimum price, and present only the items that are more expensive. (Your customers' buying criteria may vary.)
- Setting the default minimum price to 100 units of the appropriate currency. Later, we see how to allow the user to pick a minimum price.

***Example 12–9 show_product_partial.psp***

```
<%@ plsql procedure="show_product_partial" %>
<%@ plsql parameter="p_minprice" default="100" %>
<HTML>
<HEAD><TITLE>Show Items Greater Than Specified Price</TITLE></HEAD>
<BODY>
<P>This report shows the items whose price is greater than <%= p_minprice %>.
<UL>
<% FOR ITEM IN (SELECT product_name, list_price, catalog_url
                FROM product_information
                WHERE list_price > p_minprice
                ORDER BY list_price DESC)
   LOOP %>
<LI>
Item = <A HREF="<%= item.catalog_url %>"><%= item.product_name %></A><BR>
Price = <BIG><%= item.list_price %></BIG><BR>
<% END LOOP; %>
</UL>
</BODY>
</HTML>
```

After loading Example 12–9 into the database, you can pass a parameter to the `show_product_partial` procedure through a URL. The following example specifies a minimum price of 250:

```
http://www.host.com/pls/show_product_partial?p_minprice=250
```

This technique of filtering results is fine for some applications, such as search results, in which users might worry about being overwhelmed by choices. But in a retail situation, you might want to use the alternative technique illustrated in Example 12–10 so that customers can still choose to purchase other items. Note the following features of this example:

- Instead of filtering the results through a `WHERE` clause, we can retrieve the entire result set and then take different actions for different returned rows.

- We can change the HTML to highlight the output that meets their criteria. In this case, we use the background color for an HTML table row. We could also insert a special icon, increase the font size, or use some other technique to call attention to the most important rows.
- We can present the results in an HTML table.

***Example 12–10 show_product_highlighed.psp***

```
<%@ plsql procedure="show_product_highlighted" %>
<%@ plsql parameter="p_minprice" default="100" %>
<%! v_color VARCHAR2(7); %>

<HTML>
<HEAD><TITLE>Show Items Greater Than Specified Price</TITLE></HEAD>
<BODY>
<P>This report shows all items, highlighting those whose price is
 greater than <%= p_minprice %>.
<P>
<TABLE BORDER>
  <TR>
    <TH>Product</TH>
    <TH>Price</TH>
  </TR>
  <% FOR ITEM IN (SELECT product_name, list_price, catalog_url
                  FROM product_information
                  WHERE list_price IS NOT NULL
                  ORDER BY list_price DESC) LOOP
     IF item.list_price > p_minprice THEN
        v_color := '#CCCCFF';
     ELSE
        v_color := '#CCCCCC';
     END IF;
  %>
  <TR BGCOLOR="<%= v_color %>">
    <TD><A HREF="<%= item.catalog_url %>"><%= item.product_name %></A></TD>
    <TD><BIG><%= item.list_price %></BIG></TD>
  </TR>
  <% END LOOP; %>
</TABLE>
</BODY>
</HTML>
```

## Using an HTML Form to Call a PL/SQL Server Page

Example 12–11 shows a bare-bones HTML form that allows the user to enter a price. The form calls the `show_product_partial` stored procedure illustrated in Example 12–9 and passes it the entered value as the `p_minprice` parameter.

To avoid coding the entire URL of the stored procedure in the `ACTION=` attribute of the form, we can make the form a PSP file so that it resides in the same directory as the PSP file that it calls. Even though this HTML file contains no PL/SQL code, we can give it a `.psp` extension and load it as a stored procedure into the database. When the `product_form` stored procedure is executed through a URL, it displays the HTML exactly as it appears in the file.

***Example 12–11 product_form.psp***

```
<HTML>
<BODY>
```

```
<FORM method="POST" action="show_product_partial">
  <P>Enter the minimum price you want to pay:
  <INPUT type="text" name="p_minprice">
  <INPUT type="submit" value="Submit">
</FORM>
</BODY>
</HTML>
```

### Including JavaScript in a PSP File

To produce an elaborate HTML file, perhaps including dynamic content such as JavaScript, you can simplify the source code by implementing it as a PSP. This technique avoids having to deal with nested quotation marks, escape characters, concatenated literals and variables, and indentation of the embedded content.

Example 12–12 shows a version of Example 12–9 that uses JavaScript to display the order status in the browser status bar when the user moves his or her mouse over the product URL.

***Example 12–12 show_product_javascript.psp***

```
<%@ plsql procedure="show_product_javascript" %>
<%@ plsql parameter="p_minprice" default="100" %>
<HTML>
<HEAD>
  <TITLE>Show Items Greater Than Specified Price</TITLE>

<SCRIPT language="JavaScript">
<!--hide

var text=" ";

function overlink (text)
{
  window.status=text;
}
function offlink (text)
{
  window.status=text;
}

//-->
</SCRIPT>

</HEAD>
<BODY>
<P>This report shows the items whose price is greater than <%= p_minprice %>.
<P>
<UL>
<% FOR ITEM IN (SELECT product_name, list_price, catalog_url, product_status
                FROM product_information
                WHERE list_price > p_minprice
                ORDER BY list_price DESC)
   LOOP %>
<LI>
Item =
  <A HREF="<%= item.catalog_url %>"
  onMouseover="overlink('PRODUCT STATUS: <%= item.product_status %>');return true"
  onMouseout="offlink(' ');return true">
    <%= item.product_name %>
```

```
  </A>
<BR>
Price = <BIG><%= item.list_price %></BIG><BR>
<% END LOOP; %>
</UL>
</BODY>
</HTML>
```

## Debugging PL/SQL Server Page Problems

As you begin experimenting with PSP, and as you adapt your first simple pages into more elaborate ones, keep these guidelines in mind when you encounter problems:

- The first step is to get all the PL/SQL syntax and PSP directive syntax right. If you make a mistake here, the file does not compile.
  - Make sure you use semicolons to terminate lines where required.
  - If a value must be quoted, quote it. You might need to enclose a single-quoted value (needed by PL/SQL) inside double quotes (needed by PSP).
  - Mistakes in the PSP directives are usually reported through PL/SQL syntax messages. Check that your directives use the right syntax, that directives are closed properly, and that you are using the right element (declaration, expression, or code block) depending on what goes inside it.
  - PSP attribute names are case-sensitive. Most are specified in all lowercase; `contentType` and `errorPage` must be specified as mixed-case.
- When using a URL to request a PSP, you may get an error that the file is not found. In this case, note the following:
  - Make sure you are requesting the right virtual path, depending on the way the Web gateway is configured. Typically, the path includes the host name, optionally a port number, the schema name, and the name of the stored procedure (with no `.psp` extension).
  - If you use the `-replace` option when compiling the file, the old version of the stored procedure is erased. So, after a failed compilation, you must fix the error or the page is not available. You might want to test new scripts in a separate schema, then load them into the production schema.
  - If you copied the file from another file, remember to change any procedure name directives in the source to match the new file name.
  - When you get one file-not-found error, make sure to request the latest version of the page the next time. The error page might be cached by the browser. You may need to force a page reload in the browser to bypass the cache.
- When the PSP script is run, and the results come back to the browser, use standard debugging techniques to check for and correct wrong output. The difficult part is to configure the interface between different HTML forms, scripts, and CGI programs so that the right values are passed into your page. The page might return an error because of a parameter mismatch. Note the following tips:
  - To determine exactly what is being passed to your page, use `METHOD=GET` in the calling form so that the parameters are visible in the URL.
  - Make sure that the form or CGI program that calls your page passes the correct number of parameters, and that the names specified by the `NAME=` attributes on the form match the parameter names in the PSP file. If the form

includes any hidden input fields, or uses the NAME= attribute on the Submit or Reset buttons, then the PSP file must declare equivalent parameters.

- Make sure that the parameters can be cast from string into the correct PL/SQL types. For example, do not include alphabetic characters if the parameter in the PSP file is declared as a NUMBER.
- Make sure that the query string of the URL consists of name-value pairs, separated by equals signs, especially if you are passing parameters by constructing a hard-coded link to the page.
- If you are passing a lot of parameter data, such as large strings, you might exceed the volume that can be passed with METHOD=GET. You can switch to METHOD=POST in the calling form without changing your PSP file.
- Although the loadpsp command reports line numbers correctly when there is a syntax error in your source file, line numbers reported for runtime errors refer to a transformed version of the source and do not match the line numbers in the original source. When you encounter errors that produce an error trace instead of the expected Web page, you will need to locate the error through exception handlers and by printing debug output.

## Putting PL/SQL Server Pages into Production

Before putting your PSP application into production, consider issues such as usability and download speed:

- Pages can be rendered faster in the browser if the HEIGHT= and WIDTH= attributes are specified for all images. You might standardize on picture sizes, or store the height and width of images in the database along with the data or URL.
- For viewers who turn off graphics, or who use alternative browsers that read the text out loud, include a description of significant images using the ALT= attribute. You might store the description in the database along with the image.
- Although an HTML table provides a good way to display data, a large table can make your application seem slow. Often, the reader sees a blank page until the entire table is downloaded. If the amount of data in an HTML table is large, consider splitting the output into multiple tables.
- If you set text, font, or background colors, test your application with different combinations of browser color settings:
  - Test what happens if you override just the foreground color in the browser, or just the background color, or both.
  - Generally, if you set one color (such as the foreground text color), you should set all the colors through the <BODY> tag, to avoid hard-to-read combinations like white text on a white background.
  - If you use a background image, specify a similar background color to provide proper contrast for viewers who do not load graphics.
  - If the information conveyed by different colors is crucial, consider using an alternative technique. For example, you might put an icon next to special items in a table. Some users may see your page on a monochrome screen or on browsers that cannot represent different colors.
- Providing context information prevents users from getting lost. Include a descriptive <TITLE> tag for your page. If the user is partway through a procedure, indicate which step is represented by your page. Provide links to

logical points to continue with the procedure, return to a previous step, or cancel the procedure completely. Many pages might use a standard set of links that you embed using the include directive.

- In any entry fields, users might enter incorrect values. Where possible, use select lists to present a set of choices. Validate any text entered in a field before passing it to SQL. The earlier you can validate, the better; a JavaScript routine can detect incorrect data and prompt the user to correct it before they press the `Submit` button and make a call to the database.
- Browsers tend to be lenient when displaying incorrect HTML. What looks OK in one browser might look bad or might not display at all in another browser. Note the following guidelines:
  - Pay attention to HTML rules for quotation marks, closing tags, and especially for anything to do with tables.
  - Minimize the dependence on tags that are only supported by a single browser. Sometimes you can provide an extra bonus using such tags, but your application should still be usable with other browsers.
  - You can check the validity, and even in some cases the usability, of your HTML for free at many sites on the World Wide Web.

# Exhibit 9

# Oracle® Application Server Containers for J2EE

User's Guide

10*g* (9.0.4)

**Part No. B10322-01**

September 2003

ORACLE

Oracle Application Server Containers for J2EE User's Guide, 10*g* (9.0.4)

Part No. B10322-01

Copyright © 2002, 2003 Oracle Corporation. All rights reserved.

Primary Author: Sheryl Maring

Contributing Author: Brian Wright, Timothy Smith

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent and other intellectual and industrial property laws. Reverse engineering, disassembly or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

**Restricted Rights Notice** Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle is a registered trademark, and Oracle9*i* and SQL*Plus are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

# 1

# OC4J Overview

This chapter describes OC4J and demonstrates how to install OC4J with the Oracle Application Server installation.

This chapter includes the following topics:

- Introduction to OC4J
- JDK 1.4 Considerations: Cannot Invoke Classes Not in Packages
- Navigating the OC4J Documentation Set
- OC4J Installation
- Using OC4J in an Enterprise or Standalone Environment
- OC4J Communication

# Introduction to OC4J

Oracle Application Server provides a complete Java 2 Enterprise Edition (J2EE) environment written entirely in Java that executes on the Java virtual machine (JVM) of the standard Java Development Kit (JDK). You can run Oracle Application Server Containers for J2EE (OC4J) on the standard JDK that exists on your operating system. Refer to the certification matrix on `http://otn.oracle.com`.

OC4J is J2EE certified and provides all the containers, APIs, an services that J2EE specifies. OC4J is based on technology licensed from Ironflare Corporation, which develops the Orion server—one of the leading J2EE containers. Although OC4J is integrated with the Oracle Application Server infrastructure, the product and some of the documentation still contains some reference to the Orion server.

OC4J supports and is certified for the standard J2EE APIs, as listed in Table 1–1.

***Table 1–1 Oracle Application Server J2EE Support***

| J2EE 1.3 Standard APIs | Version Supported |
|---|---|
| JavaServer Pages (JSP) | 1.2 |
| Servlets | 2.3 |
| Enterprise JavaBeans (EJB) | 2.0 |
| Java Transaction API (JTA) | 1.0 |
| Java Message Service (JMS) | 1.0 |
| Java Naming and Directory Interface (JNDI) | 1.2 |
| Java Mail | 1.1.2 |
| Java Database Connectivity (JDBC) | 2.0 Extension |
| Oracle Application Server Java Authentication and Authorization Service | 1.0 |
| J2EE Connector Architecture (JCA) | 1.0 |
| JAXP | 1.1 |

The OC4J documentation assumes that you have a basic understanding of Java programming, J2EE technology, and Web and EJB application technology. This includes deployment conventions such as the `WEB-INF` and `META-INF` directories.

Examples in each of the primers assume the following:

- You have a working JDK. The JDK version 1.4.1 is installed with Oracle Application Server. However, OC4J can also work with JDK version 1.3.x.
- You have installed Oracle Application Server, including the OC4J software.
- You have started the OC4J instance.

For Web applications, when specifying a URL to execute an application in Oracle Application Server, note that by default in Oracle Application Server, you should use port 7777 to go through the Oracle HTTP Server, with OracleAS Web Cache enabled.

Examples also use standard J2EE configuration files such as `web.xml` and `application.xml`.

## JDK 1.4 Considerations: Cannot Invoke Classes Not in Packages

Among the migration considerations in moving to a Sun Microsystems JDK 1.4 environment, which is the environment that is shipped with Oracle Application Server 10*g*, there is one of particular importance to servlet and JSP developers.

As stated by Sun Microsystems, "The compiler now rejects import statements that import a type from the unnamed namespace." (This was to address security concerns and ambiguities with previous JDK versions.) Essentially, this means that you cannot invoke a class (a method of a class) that is not within a package. Any attempt to do so will result in a fatal error at compilation time.

This especially affects JSP developers who invoke JavaBeans from their JSP pages, as such beans are often outside of any package (although the JSP 2.0 specification now requires beans to be within packages, in order to satisfy the new compiler requirements). Where JavaBeans outside of packages are invoked, JSP applications that were built and executed in an OC4J 9.0.3 / JDK 1.3.1 environment will no longer work in an OC4J 9.0.4 / JDK 1.4 environment.

Until you update your application so that all JavaBeans and other invoked classes are within packages, you have the alternative of reverting back to a JDK 1.3.1 environment to avoid this issue.

**Notes:**

- The `javac -source` compiler option is intended to allow JDK 1.3.1 code to be processed seamlessly by the JDK 1.4 compiler, but this option does not account for the "classes not in packages" issue.
- Only the JDK 1.3.1 and JDK 1.4 compilers are supported and certified by OC4J. It is possible to specify an alternative compiler by adding a `<java-compiler>` element to the `server.xml` file, and this might provide a workaround for the "classes not in packages" issue, but no other compilers are certified or supported by Oracle for use with OC4J. (Furthermore, do *not* update the `server.xml` file directly in an Oracle Application Server environment. Use the Oracle Enterprise Manager.)

For more information about the "classes not in packages" issue and other JDK 1.4 compatibility issues, refer to the following Web site:

`http://java.sun.com/j2se/1.4/compatibility.html`

In particular, click the link "Incompatibilities Between Java 2 Platform, Standard Edition, v1.4.0 and v1.3".

## Navigating the OC4J Documentation Set

Most of the location of J2EE subject matter is obvious. For example, you can find out how to implement and use servlets within the *Oracle Application Server Containers for J2EE Servlet Developer's Guide*. Table 1–2 shows each J2EE subject matter and where you can find the information in the OC4J documentation set.

***Table 1–2 Location of Information for J2EE Subjects***

| J2EE Subject | The Subject is Documented in this OC4J Documentation Book |
|---|---|
| JSP | *Oracle Application Server Containers for J2EE Support for JavaServer Pages Developer's Guide* |
| JSP Tag Libraries | *Oracle Application Server Containers for J2EE JSP Tag Libraries and Utilities Reference* |
| Servlet | *Oracle Application Server Containers for J2EE Servlet Developer's Guide* |

*Table 1–2 Location of Information for J2EE Subjects*

| J2EE Subject | The Subject is Documented in this OC4J Documentation Book |
|---|---|
| EJB | *Oracle Application Server Containers for J2EE Enterprise JavaBeans Developer's Guide* |
| JTA | *Oracle Application Server Containers for J2EE Services Guide* |
| Data Source | *Oracle Application Server Containers for J2EE Services Guide* |
| JNDI | *Oracle Application Server Containers for J2EE Services Guide* |
| JMS | *Oracle Application Server Containers for J2EE Services Guide* |
| RMI and RMI/IIOP | *Oracle Application Server Containers for J2EE Services Guide* |
| Security | *Oracle Application Server Containers for J2EE Security Guide* |
| CSiV2 | *Oracle Application Server Containers for J2EE Security Guide* |
| JCA | *Oracle Application Server Containers for J2EE Services Guide* |
| Java Object Cache | *Oracle Application Server Containers for J2EE Services Guide* |
| OracleAS Web Services | *Oracle Application Server Web Services Developer's Guide* |
| HTTPS | *Oracle Application Server Containers for J2EE Services Guide* |

## OC4J Installation

OC4J is a lightweight container that is J2EE-compliant. It is configured with powerful and practical defaults and is ready to execute after installation. OC4J is installed with Oracle Application Server; therefore, see the *Oracle Application Server 10g Installation Guide* for details on OC4J installation.

## Using OC4J in an Enterprise or Standalone Environment

OC4J is installed within Oracle Application Server with the goal of managing J2EE enterprise systems. Oracle Application Server can manage multiple clustered OC4J processes. Oracle Application Server, which includes OC4J, is managed and configured through the Oracle Enterprise Manager, which can manage and configure your OC4J processes across multiple application server instances and hosts. Thus, you cannot locally manage your OC4J process using the `admin.jar` tool or by hand editing a single OC4J process' configuration files. This undermines the enterprise management provided by the Oracle Enterprise Manager.

You can still execute OC4J as you have in the past. For those who want a single OC4J instance for development environments or simple business needs, you can download OC4J in standalone mode—complete with documentation.

This following sections discusses both management options in the following sections:

- Managing Multiple OC4J Instances in an Enterprise Environment
- Managing a Single OC4J Instance

Also, the following section describes how to understand the OC4J documentation set:

- OC4J Documentation Set Assumptions

## Managing Multiple OC4J Instances in an Enterprise Environment

You manage Oracle Application Server, including OC4J, using Oracle Enterprise Manager within an enterprise system. This includes clustering, high availability, load balancing, and failover.

You configure each OC4J instance and its properties—within the context of an application server instance—using Oracle Enterprise Manager. After configuration, you start, manage, and control all OC4J instances through Oracle Enterprise Manager. You can group several OC4J processes in a cluster. You must use either the Oracle Enterprise Manager management tool or its command-line tools for starting, stopping, restarting, configuring, and deploying applications.

---

**Note:** You cannot use the OC4J standalone tool—`admin.jar`—for managing OC4J instances created in an application server instance.

You can modify the XML files locally. If you do so, you must notify Oracle Enterprise Manager that these files have been hand edited through the Distributed Configuration Management (DCM) component tool—`dcmctl`. The following is the command that you execute after hand editing an XML file:

```
dcmctl updateconfig -ct oc4j
```

DCM controls and manages configuration for Oracle Application Server instances and its Oracle HTTP Server and OC4J components. For more information on DCM, see the *Distributed Configuration Management Reference Guide*.

---

This book discusses how to start, stop, manage, and configure OC4J in an enterprise environment.

## Managing a Single OC4J Instance

You can still use a single OC4J—outside of the Oracle Application Server environment. After downloading OC4J in `oc4j_extended.zip` from OTN, you can start, manage, and control all OC4J instances through `oc4j.jar` and the `admin.jar` command-line tool. You configure either through the `admin.jar` command or by modifying the XML files by hand.

Any standalone OC4J process is not managed by Oracle Enterprise Manager and cannot be used within an Oracle Application Server enterprise environment. Typically, you would use standalone for development or for a simple single OC4J instance Web solution.

Download the *OC4J Standalone User's Guide* for information on how to start, stop, configure, and manage your standalone process.

## OC4J Documentation Set Assumptions

Aside from this book, the rest of the OC4J documentation set was written with a standalone mindset. These other books may refer to modifying XML files by hand for managing the instance. This book provides a good overview and familiarization of the Oracle Enterprise Manager configuration pages. It also guides you to understand the relationship of each Oracle Enterprise Manager page to its XML counterpart. Use the familiarity of the Oracle Enterprise Manager when reading the other OC4J books. You should be able to look at an XML representation and match it to the relevant Oracle Enterprise Manager field names.

Also, the Distributed Configuration Management (DCM) utility, `dcmctl`, provides a command-line alternative to using Oracle Enterprise Manager for some management tasks. The `dcmctl` tool uses the same distributed architecture and synchronization features as Oracle Enterprise Manager, thereby providing identical functionality in a format that is ideal for scripting and automation.

The following functions can be managed through DCM:

- administration
- managing application server instances
- managing components
- managing clusters

- deploying applications

For other DCM commands that relate to OC4J, see the *Distributed Configuration Management Reference Guide*.

# OC4J Communication

For HTTP applications, OC4J is preconfigured to execute behind the Oracle HTTP Server (OHS). You use the Oracle HTTP Server as a front-end listener and OC4J as the back-end J2EE application server.

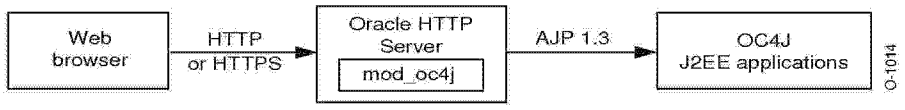
However, for RMI-based applications—such as EJB and JMS—clients should send their requests directly to OC4J. See "Understanding and Configuring OC4J Listeners" on page 3-31 for directions.

## HTTP Communication

For all incoming HTTP communication within the application server environment, you use the OHS as a front-end listener and OC4J as the back-end J2EE application server. Figure 1–1 illustrates this as follows:

1. A browser accesses the OHS listener for all HTTP requests. The Oracle HTTP Server is an Apache server. The default port number is 7777.
2. OHS, through the `mod_oc4j` module, passes the request to the OC4J server. The connection between the OHS and OC4J uses the Apache JServ Protocol (AJP) on a port number negotiated during OC4J startup. AJP is faster than HTTP, through the use of binary formats and efficient processing of message headers.

***Figure 1–1 HTTP Application Listener***




The `mod_oc4j` module is preconfigured to direct all incoming HTTP requests under the `j2ee/` Web context to OC4J. This is to separate incoming requests for JServ from those directed to OC4J. Thus, if you want to use the default routing, you can deploy your Web application into a servlet context that includes as its prefix `j2ee/`. However, any URL mapping you provide in the deployment wizard is automatically added to the `mod_oc4j` module. See "Configuring Oracle HTTP

Server With Another Web Context" on page 3-32 for information on what is added to `mod_oc4j` for you during deployment. For additional information on the `mod_oc4j` module, see the *Oracle HTTP Server Administrator's Guide*.

> **Notes:** In Oracle9*i*AS version 1.0.2.2, the default OC4J Web site did not use the Oracle HTTP Server as a front-end, and it listened using the HTTP protocol on port 8888.

## Requirements

For optimum performance, run OC4J with the JDK that is installed with Oracle Application Server Release 2, which is JDK 1.3.x.

It is not necessary to add anything to your `CLASSPATH` to run OC4J, because it loads the Java JAR and class files directly from the installation directory, from the `lib/` subdirectory, and from the deployed application EAR files.

# Exhibit 10

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

| | | |
|---|---|---|
| ORACLE CORPORATION and ORACLE U.S.A. INC., | ) | |
| Plaintiffs, | ) | |
| v. | ) | C.A. No. 06-414 (SLR) |
| EPICREALM LICENSING, LP, | ) | |
| Defendant. | ) | |

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

MORRIS, NICHOLS, ARSHT & TUNNELL LLP

*/s/ James W. Parrett, Jr.*

Mary B. Graham (#2256)
James W. Parrett, Jr. (#4292)
1201 North Market Street
P.O. Box 1347
Wilmington, DE 19899
302.658.9200
mgraham@mnat.com
jparrett@mnat.com

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

POTTER ANDERSON & CORROON LLP

*/s/ David E. Moore*

Richard L. Horwitz (#2246)
David E. Moore (#3083)
1313 N. Market Street
Hercules Plaza Sixth Floor
P.O. Box 951
Wilmington, DE 19899-0951
302.984.000
rhorwitz@potteranderson.com
dmoore@potteranderson.com

*Attorneys for epicRealm Licensig, LP*

OF COUNSEL:

James G. Gilliland
Theodore T. Herhold
Chad E. King
Robert J. Artuz
Eric M. Hutchins
TOWNSEND AND TOWNSEND AND CREW LLP
379 Lytton Avenue
Palo Alto, CA 94301
650.326.2400

OF COUNSEL:

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

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

Dated: June 30, 2008

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

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

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

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

| Claim Term | Oracle's Proposed Construction | epicRealm's Proposed Construction |
|---|---|---|
| Web server<br><br>('554 patent – claims 1, 11)<br>('335 patent – claim [1]) | HTTP-compliant server software, or a machine running such software, that receives Web page requests and returns Web pages in response to the requests | software, or a machine having software, that receives Web page requests and returns Web pages in response to the requests |
| Means for generating [said request]<br><br>('554 patent – claim 9) | § 112 ¶6 corresponding function: generating said request<br><br>structure:<br><br>a processor of a computer that is, or has, a Web client running a Web browser | "a processor of a computer that is, or has, a Web client running a Web browser" or equivalents thereof |
| Means for receiving [said request from said first computer]<br><br>('554 patent – claim 9) | § 112 ¶6 corresponding function: receiving said request from said first computer<br><br>structure:<br><br>a processor of a computer that is, or has, a Web server running a Web server executable | "a processor of a computer that is, or has, a Web server running Web server executable" or equivalents thereof |
| Page server processing means<br><br>('554 patent – claim 9) | § 112 ¶6 corresponding function: processing dynamic Web page generation requests<br><br>corresponding structure:<br><br>a processor of a computer that runs software for generating dynamic Web pages | "a processor of a computer that runs page server software (wherein page server software is page-generating software that generates a dynamic Web page)" or equivalents thereof |

2389667.1

# Exhibit 11

# Oracle® Application Server Web Cache

Administrator's Guide

10*g* Release 2 (10.1.2)

**B14046-05**

February 2006

ORACLE

Oracle Application Server Web Cache Administrator's Guide, 10*g* Release 2 (10.1.2)

B14046-05

Copyright © 2002, 2006, Oracle. All rights reserved.

Primary Author: Deborah Steiner

Contributors: Richard Anderson, Gaurav Agarwal, Guru Belur, Christine Chan, Kellepu Chander, Fang Chen, Tina Cleaveland, Rishi Divate, Joseph Errede, Jay Feenan, Patrick Fry, Priyanka GES, Ric Goell, Hideaki Hayashi, Wei Lin, Gary Ling, Peter Lubbers, Rabah Mediouni, Rajiv Mishra, Rajesh Nanda, Cyril Scott, Jiangping Shi, Michael Skarpelos, Sudheer Suggala, Parthiban Thilagar, Cathleen Trezza, Bill Wright, Zhong Xu, Naveen Zalpuri, Jean Zeng

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software—Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

This program contains third-party code from Henry Spencer and the PHP Group. Under the terms of the Henry Spencer copyright notice and the PHP license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the PHP software, and the terms contained in the following notices do not change those rights.

**Regular Expression Rights and License Notice** The Windows release of Oracle Application Server Web Cache includes source code from the regular expression (regex) directory of release 4.01 of the PHP source distribution from the PHP Group. The regex source code is a copyright of Henry Spencer and licensed from the PHP Group. The following applies only to the regex code which is compiled and linked in the webcached.exe binary.

**Henry Spencer Copyright Notice**

Copyright © 1992, 1993, 1994 Henry Spencer. All rights reserved.

This software is not subject to any license of the American Telephone and Telegraph Company or of the Regents of the University of California.

Permission is granted to anyone to use this software for any purpose on any computer system, and to alter it and redistribute it, subject to the following restrictions:

1. The author is not responsible for the consequences of use of this software, no matter how awful, even if

they arise from flaws in it.

2. The origin of this software must not be misrepresented, either by explicit claim or by omission. Since few users ever read sources, credits must appear in the documentation.

3. Altered versions must be plainly marked as such, and must not be misrepresented as being the original software. Since few users ever read sources, credits must appear in the documentation.

4. This notice may not be removed or altered.

**PHP License Agreement for regex Source Code**

Copyright © 1999, 2000 The PHP Group. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name "PHP" must not be used to endorse or promote products derived from this software without prior permission from the PHP Group. This does not apply to add-on libraries or tools that work in conjunction with PHP. In such a case, the PHP name may be used to indicate that the product supports PHP.

4. The PHP Group may publish revised and/or new versions of the license from time to time. Each version will be given a distinguishing version number. Once covered code has been published under a particular version of the license, you may always continue to use it under the terms of that version. You may also choose to use such covered code under the terms of any subsequent version of the license published by the PHP Group. No one other than the PHP Group has the right to modify the terms applicable to covered code created under this License.

# 1

# Introduction to OracleAS Web Cache

This chapter describes the performance barriers faced by Web sites and introduces the technology that can provide a complete caching solution.

This chapter contains these topics:

- What Is the Big Picture for Caching?
- Oracle's Solution to Web Site Performance Issues
- How Reverse Proxy Web Caching Works
- Benefits of Web Caching
- Features of OracleAS Web Cache
- Compatibility with Oracle Application Server Components

## What Is the Big Picture for Caching?

The Web-based computing model creates new performance challenges for application developers and administrators. To carry out e-business successfully, Web sites must protect against poor response time and system outages caused by peak loads. Slow performance translates into lower productivity and lost revenue.

Many administrators try to counter this problem by adding more **application Web servers** to their existing architecture. As more users access these Web sites, more and more servers will have to be added. In short, the hardware and manageability costs associated with adding servers often outweigh the benefits.

Static caches and content distribution services can provide some relief. However, these solutions are unable to serve content that is dynamically generated. Developers spend time writing custom caching solutions.

## Oracle's Solution to Web Site Performance Issues

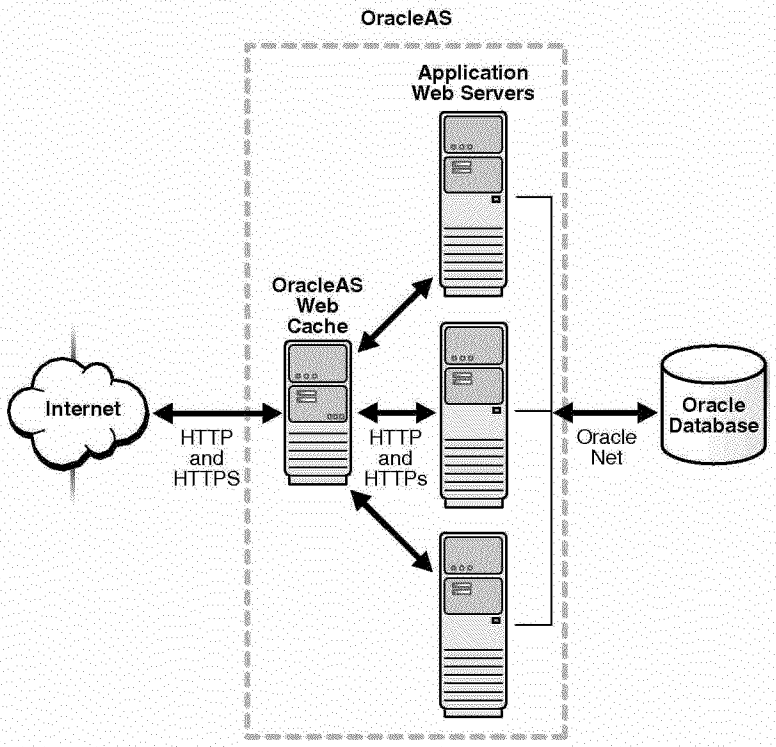
Faced with these performance challenges, e-businesses need to invest in more cost-effective technologies and services to improve the performance of their applications. Oracle offers OracleAS Web Cache to help e-businesses manage Web site and Web-based application performance issues. OracleAS Web Cache is a content-aware server accelerator, or **reverse proxy**, that improves the performance, scalability, and availability of Web sites that run on Oracle Application Server.

By storing frequently accessed URLs in memory, OracleAS Web Cache eliminates the need to repeatedly process requests for those URLs on the application Web server and database tiers. Unlike legacy proxies that handle only static objects, OracleAS Web Cache caches both static and dynamically generated content from one or more

application Web servers. Because OracleAS Web Cache is able to cache more content than legacy proxies, it provides optimal performance by greatly reducing the load on application Web server and database tiers. As an external cache, OracleAS Web Cache is also an order of magnitude faster than object caches that run within the application tier.

Figure 1–1 shows the basic architecture. OracleAS Web Cache sits in front of application Web servers, caching their content, and providing that content to clients that request it. When Web browsers access the Web site, they send **HTTP protocol** or **HTTPS protocol** requests to OracleAS Web Cache. OracleAS Web Cache, in turn, acts as a virtual server on behalf of the application Web servers. If the requested content has changed, OracleAS Web Cache retrieves the new content from the application Web servers. The application Web servers may retrieve their content from an Oracle database. OracleAS Web Cache can be deployed on its own dedicated tier of computers or on the same computer as the application Web servers.

***Figure 1–1 OracleAS Web Cache Architecture***




**Note:** OracleAS Web Cache is compatible with Oracle HTTP Server or any other HTTP-compliant application Web server. See Appendix D for further information about third-party application Web server support.

## How Reverse Proxy Web Caching Works

As a reverse proxy, OracleAS Web Cache acts a gateway to the backend servers. Figure 1–2 on page 1-3 shows how OracleAS Web Cache acts a reverse proxy cache for a backend application Web server. OracleAS Web Cache has an IP address of `144.25.190.241` and the application Web server has an IP address of



ORCL00513910

144.25.190.242. The steps for browser interaction with OracleAS Web Cache follow:

1. A browser sends a request to a Web site named www.company.com:80.

   This request in turn generates a request to Domain Name System (DNS) for the IP address of the Web site.

2. DNS returns the IP address of the load balancer for the site, that is, 144.25.190.240.

3. The browser sends the request for a Web page to the load balancer. In turn, the load balancer sends the request to OracleAS Web Cache server 144.25.190.241.

4. If the requested content is in its cache, then OracleAS Web Cache sends the content directly to the browser. This is called a **cache hit**.

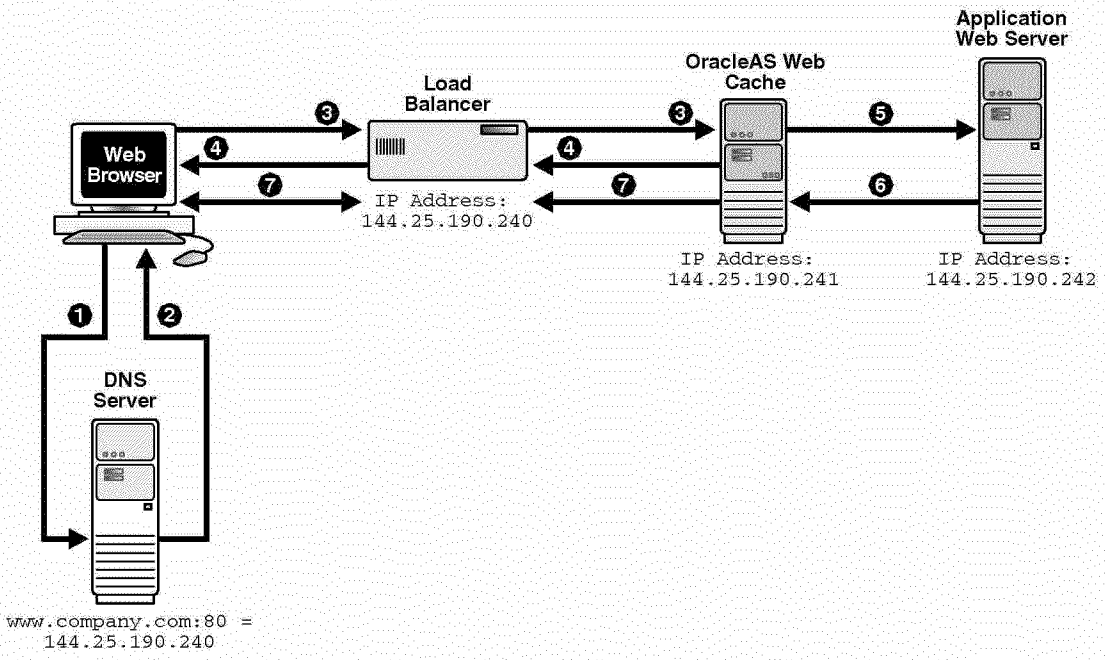
5. If OracleAS Web Cache does not have the requested content or the content is stale or invalid, it hands the request off to application Web server 144.25.190.242. This is called a **cache miss**.

6. The application Web server sends the content to OracleAS Web Cache.

7. OracleAS Web Cache sends the content to the client and stores a copy of the page in cache.

> **Note:** OracleAS Web Cache marks an object stored in the cache for removal when it becomes invalid or outdated, as described in "About Cache Consistency and Performance Assurance" on page 2-1.

***Figure 1–2 Web Server Acceleration***




If you do not require caching support, you can configure OracleAS Web Cache solely as a software load balancer or reverse proxy. You configure OracleAS Web Cache settings for this mode.

**See Also:** "OracleAS Web Cache Solely as a Software Load Balancer or Reverse Proxy" on page 8-42

## Benefits of Web Caching

Web caching provides the following benefits for Web-based applications:

- Performance

  Running on inexpensive hardware, OracleAS Web Cache can increase the throughput of a Web site by several orders of magnitude. In addition, OracleAS Web Cache significantly reduces response time to client requests by storing objects in memory and by serving compressed versions of objects to clients that support the GZIP encoding method.

- Scalability

  In addition to unparalleled throughput, OracleAS Web Cache can sustain thousands of concurrent client connections, meaning that visitors to a site see fewer application Web server errors, even during periods of peak load.

- High availability

  OracleAS Web Cache supports load balancing and failover detection for application Web servers. These features ensure that cache misses are directed to the most available, highest-performing Web server in the server farm. Moreover, a patent-pending capacity heuristic guarantees performance and provides surge protection when the application Web server load increases.

- Cost savings

  Better performance, scalability and availability translates into cost savings for Web site operators. Because fewer application Web servers are required to meet the challenges posed by traffic spikes and denial of service attacks, OracleAS Web Cache offers a simple and inexpensive means of reducing a Web site's cost for each request.

- Developer productivity

  Application developers can use OracleAS Web Cache to cache content rather than design and develop application-specific caches.

## Features of OracleAS Web Cache

The features of OracleAS Web Cache make it a perfect caching service for e-business Web sites that host online catalogs, news services, and portals. These features include:

- Whole-Page Caching for Static and Dynamic Content Caching
- Cache Invalidation and Expiration
- Performance Assurance
- Virtual Hosting
- Cache Clustering
- Cache Hierarchies
- Origin Server Surge Protection, Load Balancing, Failover, and Session Binding
- Security



ORCL00513912

- Compression
- Auto-Restart

## Whole-Page Caching for Static and Dynamic Content Caching

Caching rules determine which objects OracleAS Web Cache caches. Rules fall into three categories:

- Rules for static content, such as GIF, JPEG, or static HTML files.
- Rules for dynamically generated content created using technologies like JavaServer Pages (JSP), Active Server Pages (ASP), PL/SQL Server Pages (PSP), Java servlets, PHP Hypertext Preprocessor (PHP), and Common Gateway Interface (CGI). Support of these technologies enables OracleAS Web Cache to recognize rules for the following:
  - Multiple-version objects for the same URL, that is, the same URL with slightly different content
  - Session-aware rules for pages containing session information
  - Personalization rules for pages containing personalized greetings, such as "Welcome *Name*," and **session-encoded URLs**
- Pages that require personalized content assembly of dynamic **Edge Side Includes (ESI)** fragments.

**See Also:**

- "How OracleAS Web Cache Can Cache Dynamically Generated Content" on page 2-3 for further information about dynamically-generated content
- "About Edge Side Includes (ESI) for Partial Page Caching" on page 2-13

## Cache Invalidation and Expiration

OracleAS Web Cache supports **invalidation** as a mechanism to keep the cache consistent with the content on the application Web servers. Administrators can invalidate cache content in one of three ways:

- Manual methods, including OracleAS Web Cache Manager or `telnet`
- Automatic methods with database triggers, scripts, or application logic
- Inline invalidation implemented as part of ESI template pages

When objects are invalidated and a client requests them, OracleAS Web Cache refreshes them with new content from the application Web server.

In addition to invalidation, OracleAS Web Cache provides **expiration**, whereby an expiration time limit is assigned to the objects. When an object expires, OracleAS Web Cache treats it like an invalid object. If the object is requested by a client, OracleAS Web Cache refreshes it with updated content from the application Web server.

**See Also:** "About Cache Consistency and Performance Assurance" on page 2-1 for further information about invalidation and expiration



ORCL00513913

## Performance Assurance

When a large number of objects have been invalidated, the retrieval of a new object can result in overburdened application Web servers.

To handle performance issues while maintaining cache consistency, OracleAS Web Cache uses built-in **performance assurance heuristics** that enable it to assign a queue order to objects. These heuristics determine which objects OracleAS Web Cache can serve stale and which objects to refresh immediately. OracleAS Web Cache refreshes objects with a higher priority first and objects with a lower priority at a later time.

OracleAS Web Cache bases the queue order of objects on **popularity** and **validity** assigned during invalidations. If the current load and capacity of the application Web server is not exceeded, then OracleAS Web Cache refreshes the most popular and least valid objects first.

> **See Also:** "About Cache Consistency and Performance Assurance" on page 2-1 for further information about performance assurance

## Virtual Hosting

OracleAS Web Cache caches and assembles dynamic content for one or more Web sites. From the perspective of OracleAS Web Cache, a site can be either a **virtual host site** or an **ESI provider site**. Depending on the site category, you can configure OracleAS Web Cache to perform different functions.

This section covers the following topics:

- Virtual Host Sites
- ESI Provider Sites
- Site Definitions and Site-to-Server Mappings
- How OracleAS Web Cache Locates Application Web Severs or Proxy Servers

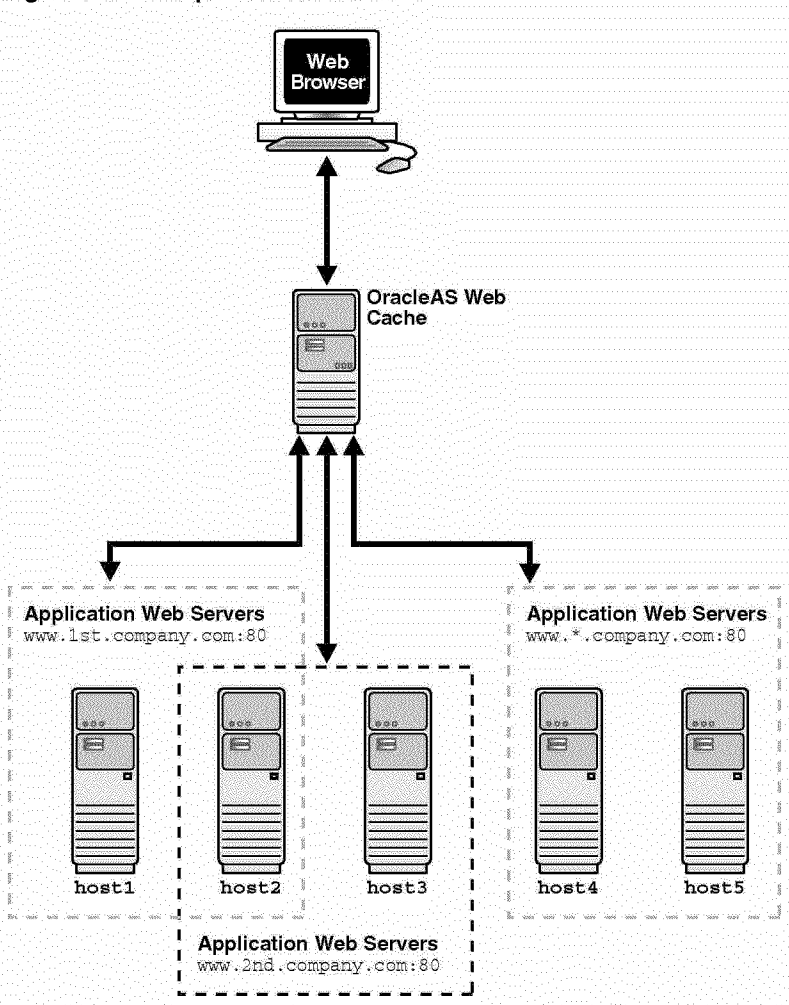
### Virtual Host Sites

Virtual host sites are sites hosted by OracleAS Web Cache. In other words, clients can request cached content from these sites through OracleAS Web Cache. Figure 1–3 on page 1-7 shows OracleAS Web Cache caching content for two sites, `www.1st.company.com` and `www.2nd.company.com`. An additional mapping of `www.*.company.com` uses *, enabling additional virtual sites that map to `host4` and `host5` to be added. In addition to caching content, OracleAS Web Cache can also assemble ESI fragments from these sites.



ORCL00513914

*Figure 1–3 Multiple Virtual Host Sites*




## ESI Provider Sites

ESI provider sites are those sites that OracleAS Web Cache contacts for ESI assembly only. Browsers are not allowed to request content from these sites.

Figure 1–4 on page 1-8 shows an ESI provider site configuration. In this configuration, OracleAS Web Cache receives a request for a page with ESI markup tags. OracleAS Web Cache sends the request to the application Web server. The application Web server uses an application to create a template page and sends it back to OracleAS Web Cache for assembly. OracleAS Web Cache includes the ESI fragments for the template page either with a cached copy of the contents or by contacting the following for that fragment's contents:

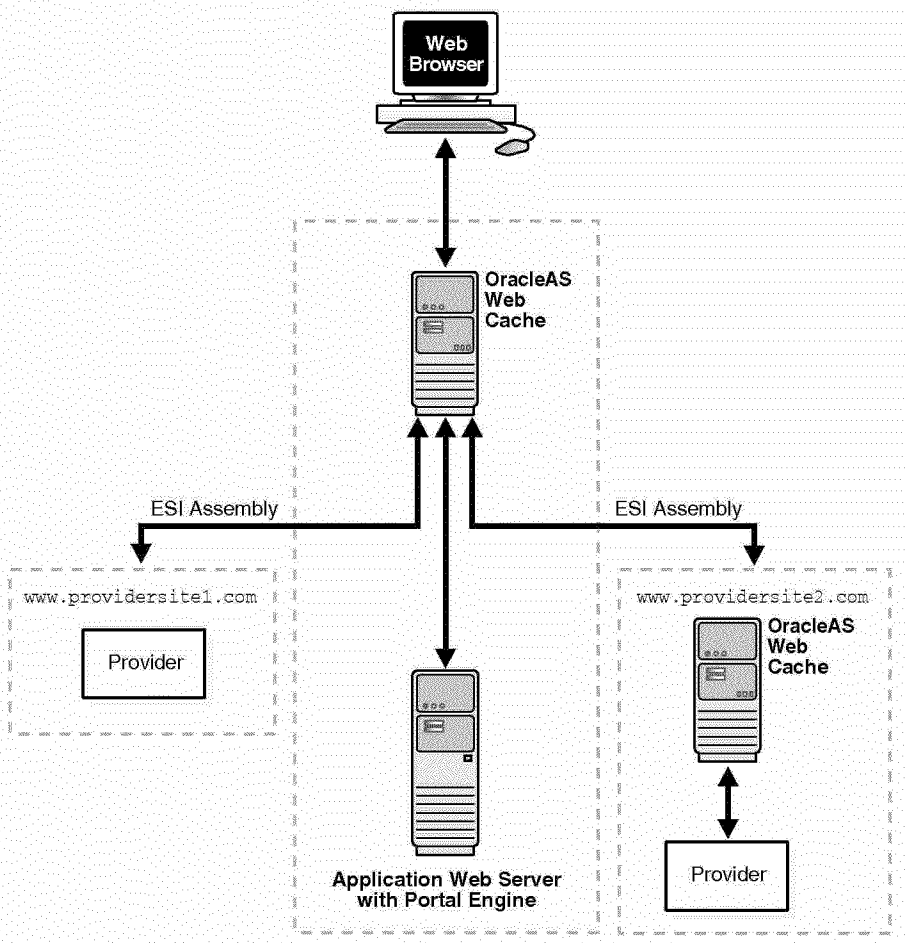
- Provider site `www.providersite1.com`
- Another OracleAS Web Cache server, which is caching content for `www.providersite2.com`

> **See Also:** "About Edge Side Includes (ESI) for Partial Page Caching" on page 2-13 for further information about ESI



ORCL00513915

*Figure 1–4 Multiple ESI Provider Sites*




## Site Definitions and Site-to-Server Mappings

In order for OracleAS Web Cache to recognize a virtual host site or an ESI provider site, administrators need to perform the following steps:

1. Create a site definition that includes all the names and listening port numbers of the site that clients specify.

   If two sites share the same host name, specify a URL path prefix to distinguish the sites.

2. Determine if OracleAS Web Cache communicates with application Web servers or **proxy servers**.

   OracleAS Web Cache uses application Web servers for internal sites and proxy servers for external sites outside a firewall.

3. Specify the host name and listening port number of the application Web server or proxy server.

4. To ensure requests are sent to the appropriate server, map the site to the application Web servers or proxy servers.

5. Order the mappings. For each request, the first matching mapping is used.

   - Because mappings that use the wildcard * encompass a broader scope, give these mappings a lower priority than other mappings.



ORCL00513916

- Because requests are resolved to the first matching mapping, give mappings that contain the optional URL path prefix a higher priority than those mappings without an URL path prefix.

  For example, you should order the following mappings as follows:

  ```
  http://www.company.com/portal/page?_pageid=33,4232&_dad=portal
  http://www.company.com/um/traffic_cop?mailid=inbox
  http://www.company.com
  ```

  If you instead reorder the mappings as follows, the request for URLs `http://www.company.com/portal/page?_pageid=33,4232&_dad=portal` and `http://www.company.com/um/traffic_cop?mailid=inbox` will never be reached. Requests for these URLs will instead resolve to `http://www.company.com` because it is listed first:

  ```
  http://www.company.com
  http://www.company.com/portal/page?_pageid=33,4232&_dad=portal
  http://www.company.com/um/traffic_cop?mailid=inbox
  ```

6. Create caching rules that apply to the site and global rules that apply to all sites.

   The site-specific caching rules are given a higher priority than the global rules.

To further understand the mappings, reconsider Figure 1–3 on page 1-7. Web sites `www.1st.company.com:80` and `www.2nd.company.com:80` can have site aliases of `1st.company.com:80` and `2nd.company.com:80`, respectively. The site to application Web server mappings are as follows:

- `www.1st.company.com` maps to application Web servers `host1` and `host2`
- `www.2nd.company.com` maps to application Web servers `host2` and `host3`
- `www.*.company.com` maps to `host4` and `host5`

### How OracleAS Web Cache Locates Application Web Severs or Proxy Servers

When OracleAS Web Cache receives a client request for an object, it determines the destination site using one of the following elements:

- `Host` request-header field from the request
- Host portion of the requested URL
- `src` attribute of the ESI `<esi:include>` tag

OracleAS Web Cache then looks up the configured site settings and mappings to determine if the site is supported, and the origin servers and caching rules for the site. If there are no site settings and mappings for external ESI provider sites, OracleAS Web Cache uses Domain Name System (DNS) to resolve the site name.

If the request does not include host information, then OracleAS Web Cache sends the request to the default site. The default site definition includes the site host name and port information. After an Oracle Application Server installation, a default site definition that includes the host name and listening port of the Oracle HTTP Server is established. If you choose to configure another site as the default site that includes a URL path prefix, the prefix is excluded in matching requests to the site. For example, if you specify a default site of `www.company.com` and a URL path prefix of `/portal`, requests are matched for `www.company.com`, not `www.company.com/portal`.



ORCL00513917

**See Also:**

- Chapter 8 for configuration details
- "Default Site Example" on page 8-28 for default site settings

## Cache Clustering

To increase the availability and scalability of your Web site, you can configure multiple instances of OracleAS Web Cache to run as members of a **cache cluster**. A cache cluster is a loosely coupled collection of cooperating OracleAS Web Cache cache instances working together to provide a single logical cache.

Cache clusters provide failure detection and failover of caches, increasing the availability of your Web site. If a cache fails, other members of the cache cluster detect the failure and take over ownership of the cached content of the failed cluster member.

By distributing the Web site's content across multiple Web caches, more content can be cached and more client connections can be supported, expanding the capacity of your Web site.

**See Also:**

- Chapter 3 for more information about cache clusters
- Chapter 10 for configuration details

## Cache Hierarchies

Many Web-based applications mirror their Web sites in strategic geographical locations. You can deploy multiple instances of OracleAS Web Cache in a **cache hierarchy** so that a local cache stores content from a cache in a central data center for a local market. This enables caches serving local requests to shorten response time and reduce bandwidth and rack space costs for the content provider.

OracleAS Web Cache provides supports two kinds of cache hierarchies:

- A **distributed cache hierarchy** in which a **central cache** acts as an application Web server to a **remote cache**.
- An **ESI cache hierarchy** in which a **provider cache** acts as an application Web server to a **subscriber cache**.

In place of an individual OracleAS Web Cache server in a hierarchy, you can deploy a cache cluster.

A distributed cache hierarchy centralizes the management of application logic and data to the central cache and provides remote assembly and delivery of content. Compared with full-scale mirroring and database replication, a distributed cache hierarchy provides a more cost-effective model of distributed computing.

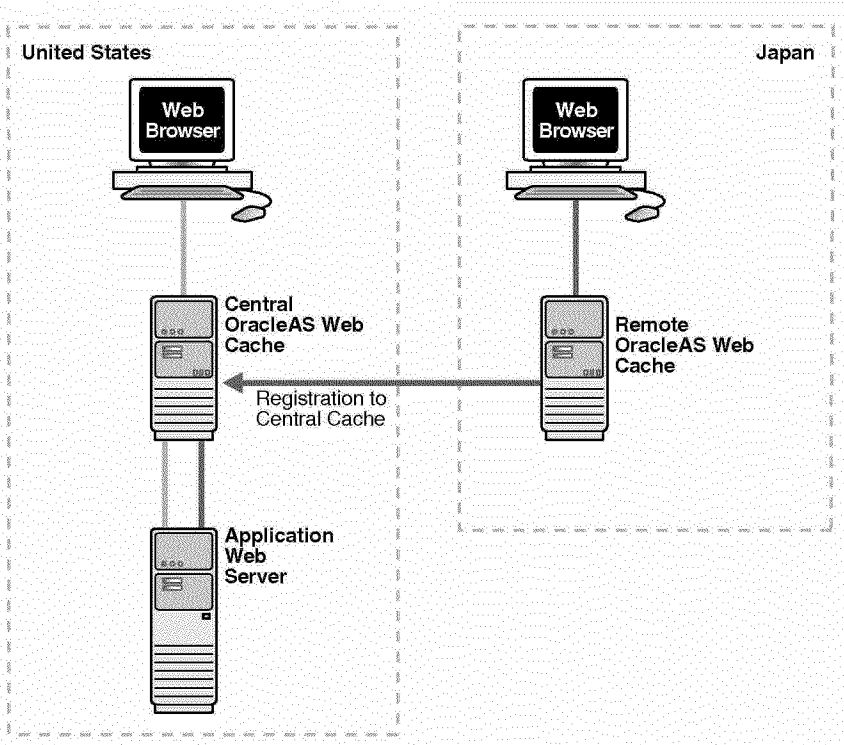
The remote cache is configured with the central cache as its application Web server. When the remote cache requests content from the central cache to serve a request, the remote cache identifies itself as an OracleAS Web Cache to the central cache during a transparent registration process. After registration is complete, the central cache establishes a hierarchical relationship with the remote cache. Registration also enables invalidation messages to be propagated from the central cache to the remote cache.

Figure 1–5 on page 1-11 depicts a distributed cache hierarchy. A central cache resides in the United States office and a remote cache resides in Japan. While the central cache in United States stored content retrieved from an application Web server, the remote cache in Japan stores retrieved content from the central cache.



ORCL00513918

*Figure 1–5 Distributed Cache Hierarchy*



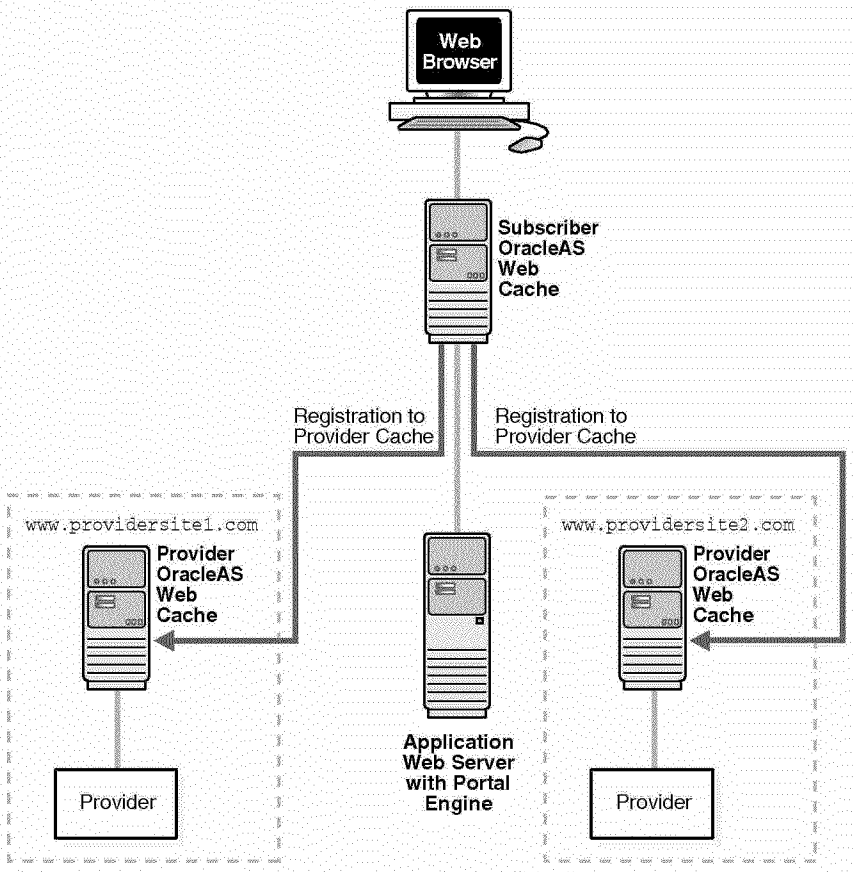

In an ESI cache hierarchy, the subscriber cache is configured with the provider caches as its application Web servers. When the subscriber cache requests content from the provider caches for ESI assembly, the subscriber cache identifies itself as an OracleAS Web Cache server to the provider caches during a transparent registration process. After registration is complete, the provider caches establish a hierarchical relationship with the subscriber cache. Registration also enables invalidation messages to be propagated from the provider caches to the subscriber cache.

Figure 1–6 on page 1-12 depicts an ESI cache hierarchy. A subscriber cache performs ESI assembly. Provider caches locally store ESI fragments for **ESI provider sites** `www.providersite1.com` and `www.providersite2.com`. During ESI page assembly, the subscriber cache contacts the provider caches for the ESI fragments. By storing the ESI fragments locally on the provider caches, fragments are stored both by the provider and subscriber caches. This provides for quick page assembly.



ORCL00513919

*Figure 1–6 ESI Cache Hierarchy*




**See Also:**

- "Invalidation in Hierarchies" on page 13-2 for information about how invalidation messages are propagated in a hierarchy of OracleAS Web Cache servers
- Chapter 11 for configuration details

## Origin Server Surge Protection, Load Balancing, Failover, and Session Binding

OracleAS Web Cache provides the following features for the application Web server and proxy server it supports:

- Surge Protection
- Stateless Load Balancing
- Backend Failover
- Session Binding (Stateful Load Balancing)

Where applicable, the term **origin server** is used in place of application Web server or proxy server to simplify the concepts presented in this section.

### Surge Protection

OracleAS Web Cache passes requests for non-cacheable, stale, or missing objects to origin servers. To prevent an overload of requests on the origin servers, OracleAS Web Cache has a surge protection feature that enables you to set a limit on the number of



ORCL00513920

concurrent requests that the origin servers can handle. When the limit is reached, subsequent requests are queued. If the queue is full, then OracleAS Web Cache rejects the request and serves a site busy error page to the client that initiated the request.

### Stateless Load Balancing

Most Web sites are served by multiple origin servers running on multiple computers that share the load of HTTP and HTTPS requests. All requests that OracleAS Web Cache cannot serve are passed to the origin servers. OracleAS Web Cache balances the load among origin servers by determining the percentage of the available **capacity**, the **weighted available capacity** of each origin server. OracleAS Web Cache sends a request to the origin server with the most weighted available capacity. The weighted available capacity is determined by the following formula:

```
(Capacity - Load) / Capacity
```

where:

- `Capacity` is the maximum number of concurrent connections that the origin server can accept
- `Load` is the number of connections currently in use

If the weighted available capacity is equal for multiple origin servers, OracleAS Web Cache sends requests to the origin servers using **round robin**. With round robin, the first origin server in the list of configured servers receives the request, then the second origin server receives the second request. If the weighted available capacity is not equal, OracleAS Web Cache sends the request to the origin server with the most available capacity.

If the load of origin servers is equivalent, OracleAS Web Cache continues to use round robin, even when capacity is not equal for origin servers. Therefore, it is possible to see an even distribution of requests to origin server when the capacities are not configured to be the same.

To configure load balancing for a site, set the capacity of each origin server, and create *one* site-to-server mapping that maps *all* the applicable origin servers to the site.

**See Also:**

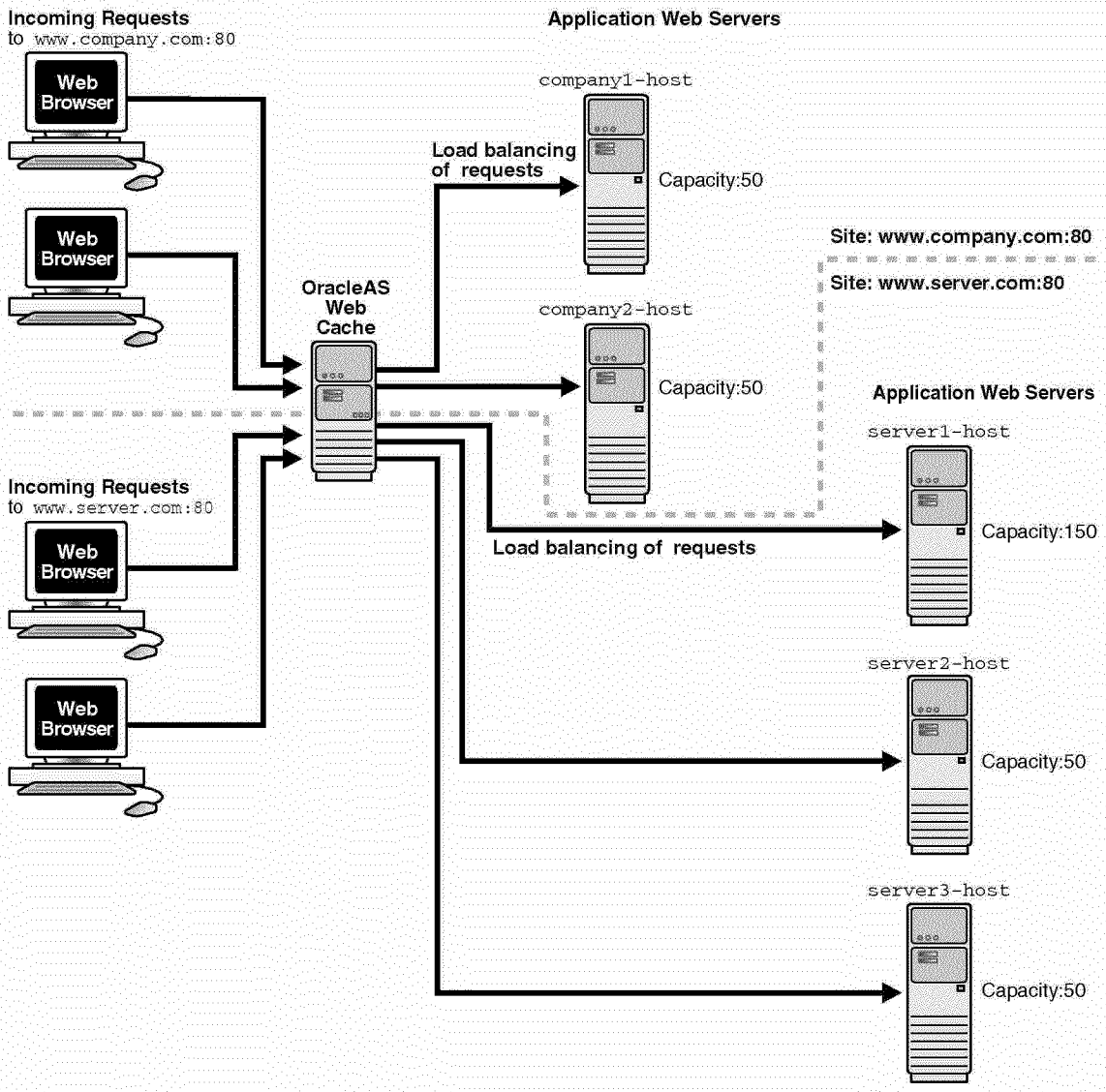
- "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25 for instructions on specifying capacity
- "Task 10: Configure Web Site Settings" on page 8-27 for instructions on creating site-to-server mappings

Figure 1–7 on page 1-14 shows two sites, `www.company.com:80` and `www.server.com:80`. The site `www.company.com:80` is supported by application Web servers `company1-host` and `company2-host` with capacities of 50 each. The site `www.server.com:80` is supported by application Web servers `server1-host`, `server2-host`, and `server3-host` with capacities of 150, 50, and 50, respectively.



ORCL00513921

*Figure 1–7 Load Balancing*




Assuming all application Web servers have an initial load of 0, the requests to `www.company.com:80` and `www.server.com:80` will be distributed in the following manner:

- The requests to `www.company.com:80` are distributed between the two origin servers using round robin.

  The requests to `company1-host` and `company2-host` will be distributed between the two origin servers so that they maintain an equal load. The first request is sent to `company1-host`. The second request is sent to `company2-host` if `company1-host` is still processing the first request. The third and subsequent requests are sent to the origin server that has the highest weighted available capacity.

  When the capacities are equal, OracleAS Web Cache uses round robin to distribute requests.



ORCL00513922

- The requests to `www.server.com:80` are distributed between three origin servers using the weighted available capacity percentage.

  The first request to `www.server.com:80` is sent to `server1-host`, because it is the first in the configured list. The second request is sent to `server2-host`, because `server1-host` is still processing the first request and has a weighted available capacity of 99.3 percent and `server2-host` has a weighted available capacity of 100 percent. The third request is sent to `server3-host` because `server2-host` is still processing a request and has a weighted available capacity of 98 percent and `server3-host` has a weighted available capacity of 100 percent. The fourth request is sent to `server1-host` because `server2-host` and `server3-host` are still processing requests and have weighted available capacities of 98 percent. The fifth request is sent to `server1-host` because its weighted available capacity is 98.6 percent, which is still greater than `server2-host` and `server3-host`, respectively.

  When the capacities and loads are not equal, OracleAS Web Cache uses the weighted available capacity to distribute requests. If requests were processed before new requests came in, then it is possible for all three origin servers to have loads of 0. In this case, OracleAS Web Cache uses round robin.

If you do not require caching support and need a low-cost solution to a hardware load balancer, you can configure OracleAS Web Cache solely as a software load balancer. This configuration mode is useful for managing traffic to a low-volume, departmental, or test Web site.

**See Also:** "OracleAS Web Cache Solely as a Software Load Balancer or Reverse Proxy" on page 8-42

### Backend Failover

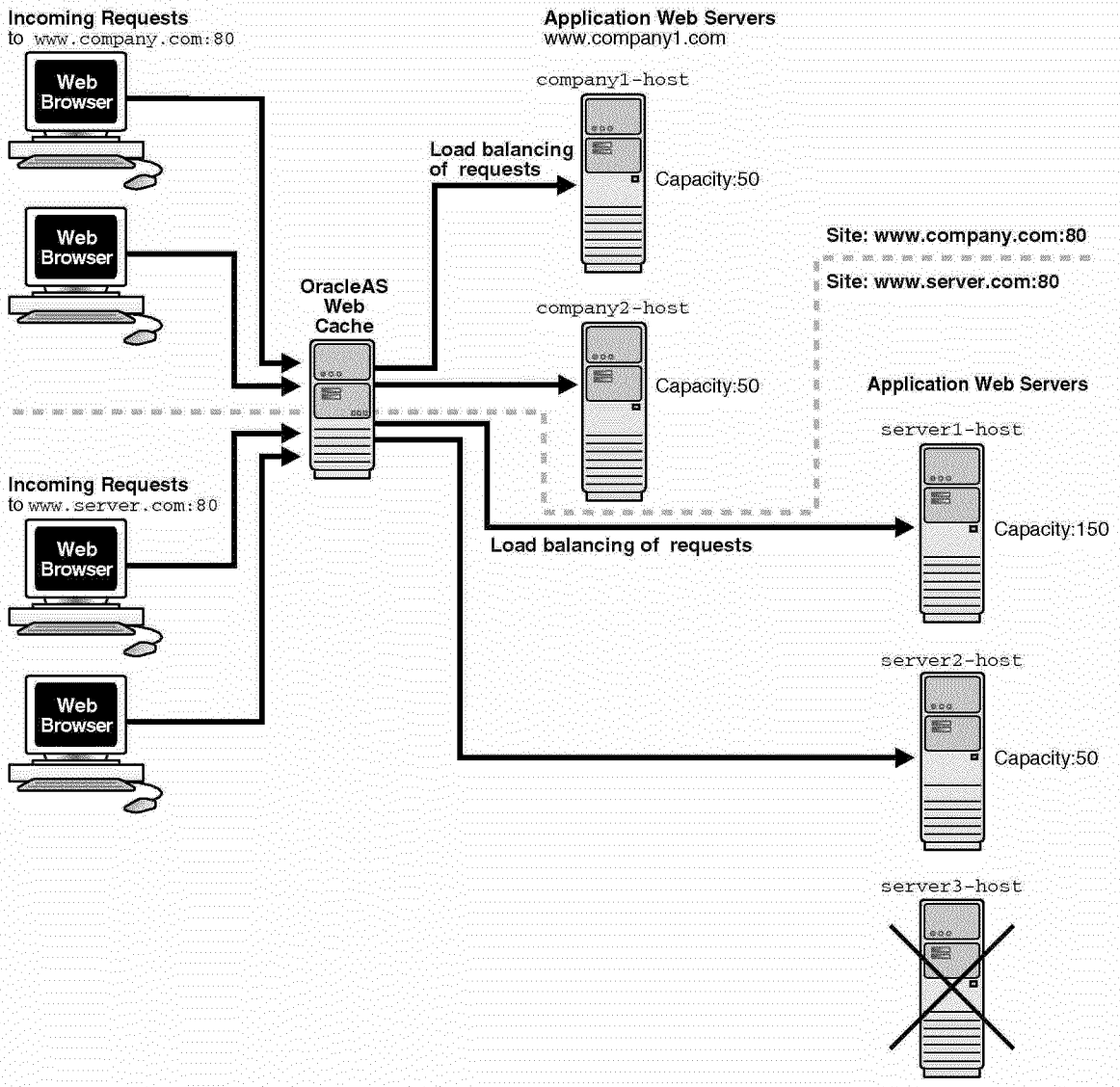
After a specified number of continuous request failures, OracleAS Web Cache considers an origin server as failed. When an origin server fails, OracleAS Web Cache automatically distributes the load over the remaining origin servers and polls the failed origin server for its current up or down status until it is back online. Existing requests to the failed origin server result in errors. However, new requests are directed to the other origin servers. When the failed server returns to operation, OracleAS Web Cache includes it in its weighted available capacity to load balance requests.

The **failover** feature is shown in Figure 1–8 on page 1-16. An outage of `server3-host`, which had a capacity of 50, results in 75 percent of requests being distributed to `server1-host` and 25 percent request being distributed to `server2-host`.



ORCL00513923

*Figure 1–8 Failover*




## Session Binding (Stateful Load Balancing)

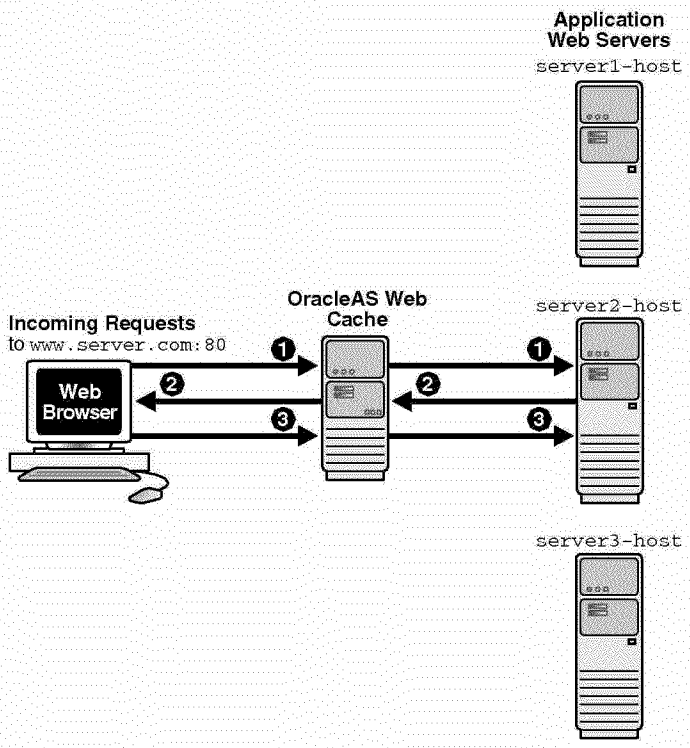
OracleAS Web Cache supports applications that use a session ID or **session cookie** to bind user sessions to a given origin server in order to maintain state for a period of time. To utilize the **session binding** feature, the origin server itself must maintain state, that is, it must be stateful. An application binds user sessions by including session data in the HTTP header or body it sends to a client in such a way that the client is forced to include it with its next request. This data is transferred between the origin server and the client through OracleAS Web Cache either with an **embedded URL parameter** or through a cookie, which is a text string that is sent to and stored on the client. OracleAS Web Cache does not process the value of the parameter or cookie; it simply passes the information back and forth between the origin server and the client.

Figure 1–9 on page 1-17 shows how OracleAS Web Cache supports objects that use session binding.



ORCL00513924

*Figure 1–9 Session Binding*




The steps for how session binding works for requests are as follows:

1. When a request first comes in, OracleAS Web Cache uses load balancing to determine to which origin server the request is forwarded. In this example, application Web server `www.server2.com` is selected.
2. If the requested object requires session binding, the origin server sends the session information back to the client through OracleAS Web Cache in the form of a cookie or an embedded URL parameter.
3. OracleAS Web Cache sends subsequent requests for the session to the origin server that established the session, bypassing load balancing. In this example, application Web server `www.server2.com` handles the subsequent requests.

To configure session binding:

1. Specify a session definition that specifies the name of session cookie or embedded URL parameter.

   You can also choose to use one of the default definitions, described in row Session Definitions of Table 8–1 on page 8-1.

2. Specify session binding settings.

In a cache cluster, all cache cluster members must be able to determine which origin server established the session, although the request was routed originally through only one cache cluster member. To configure session binding in a cache cluster, you must enable the Session Binding cookie, which is specific to OracleAS Web Cache. OracleAS Web Cache generates the cookie, which tracks session information so that subsequent requests are bound to the origin server that established the session.



ORCL00513925

**See Also:** "Bind a Session to an Origin Server" on page 8-39 for instructions on configuring session binding

**Note:** If an origin server cannot accept any more connections because of the load, OracleAS Web Cache disables session binding to that origin server and attempts to connect to another origin server.

## Security

**See:** Chapter 4, "OracleAS Web Cache Security"

## Compression

You can specify that OracleAS Web Cache compress both cacheable and non-cacheable objects with either OracleAS Web Cache Manager or the `compress` control directive of the `Surrogate-Control` response-header field.

OracleAS Web Cache correctly handles compression of different types of content and different types of browsers. For example, OracleAS Web Cache never compresses GIF or ZIP files even if you enable compression. Similarly, OracleAS Web Cache does not compress cascading style sheets for the Netscape 4.*x* or Internet Explorer 5.5 browsers. With these browsers, compressed cascading style sheets can cause background attributes, such as background images, to not appear in the output.

Because compressed objects are smaller in size, they are delivered faster to browsers with fewer round-trips, reducing overall **latency**. Compressed content is then expanded by browsers that support the GZIP compression in the `Accept-Encoding` request-header field.

In a cache hierarchy, the cache closest to the browser, a remote cache in a distributed cache hierarchy or a subscriber cache in an ESI cache hierarchy, does not perform any compression actions on objects. Instead, you configure the cache closest to the origin server, a central cache in a distributed cache hierarchy or a provider cache in an ESI cache hierarchy, to compress non-cacheable objects. By configuring compression on non-cacheable responses from downstream caches, you can ensure content is delivered more effectively to browsers.

Generally, the remote cache in a distributed cache hierarchy or the subscriber cache in an ESI cache hierarchy does not forward the `Accept-Encoding` request-header field to the origin server. However, if a browser request is for a non-cacheable object, the remote or subscriber cache can send this header to the central or provider cache. In turn, these caches instead compress the object.

On average, OracleAS Web Cache is able to compress text files by a factor of 4. For example, 300 KB files are compressed down to 75 KB.

**See Also:**

- "Configuring Caching Rules and Rule Association" on page 12-7 for instructions on configuring compression
- "Using the Surrogate-Control Response Header as an Alternative to Caching Rules" on page 12-30 for instructions on configuring the `Surrogate-Control` response-header field



ORCL00513926

### Auto-Restart

OracleAS Web Cache provides an auto-restart mechanism that checks that the cache is running and automatically restarts the cache if it is not running.

> **See Also:** "Task 3: Configure Auto-Restart Settings" on page 8-10 for instructions on enabling or disabling auto-restart

## Compatibility with Oracle Application Server Components

Table 1–1 describes OracleAS Web Cache compatibility with other Oracle Application Server components.

***Table 1–1 Compatibility with Other Oracle Application Server Components***

| Component | Description |
|---|---|
| Oracle Business Intelligence Discoverer | OracleBI Discoverer is closely integrated with OracleAS Web Cache to improve Discoverer Viewer's overall scalability, performance, and availability. OracleBI Discoverer uses ESI `Surrogate-Control` headers to govern cacheability of other non-configured responses. Because of this integration, the load on mid-tier and database servers in OracleBI Discoverer deployments is reduced, more Discoverer Viewer users are able to access the system concurrently, and those users experience significantly better response times for workbook operations and common business intelligence queries.<br>**See Also:** *Oracle Business Intelligence Discoverer Configuration Guide* |
| Oracle Application Server Forms Services | You can deploy OracleAS Web Cache as a load balancer with OracleAS Forms Services applications.<br>**See Also:** *Oracle Application Server Forms Services Deployment Guide* |
| Oracle Application Server Portal | OracleAS Web Cache has been closely integrated with OracleAS Portal to improve its overall scalability, performance, and availability. OracleAS Portal ships with a number of pre-defined caching and invalidation policies that ensure optimal use of OracleAS Web Cache. OracleAS Web Cache controls have been built into the OracleAS Portal administrative user interface and can also be specified by content providers through the Portal Developer Kit (PDK).<br>**See Also:** Sections 5.7, "Configuring OracleAS Web Cache Caching in OracleAS Portal" and 5.8 "Configuring OracleAS Portal to Use a Dedicated OracleAS Web Cache Instance," in *Oracle Application Server Portal Configuration Guide* for configuration details |
| Oracle Application Server Single Sign-On | Applications that use OracleAS Single Sign-On can use OracleAS Web Cache. You can configure OracleAS Web Cache to cache content for Oracle HTTP Servers running Single Sign-On partner applications. While you should not cache content from OracleAS Single Sign-On servers for security reasons, you can configure OracleAS Web Cache as a software load balancer in front of multiple Single Sign-On mid-tiers.<br>**See Also:**<br>▪ "Leveraging Oracle Identity Management Infrastructure" on page 4-7<br>▪ *Oracle Application Server Single Sign-On Administrator's Guide* |



ORCL00513927

*Table 1–1 (Cont.) Compatibility with Other Oracle Application Server Components*

| Component | Description |
|---|---|
| Oracle Application Server Wireless | OracleAS Wireless is integrated with OracleAS Web Cache to improve page rendering performance and scalability. It should be noted that OracleAS Web Cache does not understand WAP and is not used by OracleAS Wireless in the traditional sense in that the cache does not "front-end" the wireless server. Instead, the cache is used as a repository for post-transformed content; the wireless runtime determines what content needs to be inserted into the cache and when to expire content in the cache. OracleAS Web Cache, in this case, acts as a device adaptation cache rather than a reverse-proxy cache. Since markup content is cached using OracleAS Web Cache, the performance and scalability benefits are due to two factors—reduced device adaptation costs and significantly reduced adapter invocation costs. The savings in terms of device adaptation costs stem from the fact that content that can be shared across users and sessions is essentially transformed only once (for each logical device) from its Mobile XML format. Secondly, since the content is not generated every time by an adapter, the total adapter invocation cost is significantly reduced for a site that has a large subset of cacheable pages.<br>**See Also:** *Oracle Application Server Wireless Administrator's Guide* |



ORCL00513928

# 3

# Cache Clustering

You can configure multiple instances of OracleAS Web Cache to run as independent caches, with no interaction with one another. Most of the deployment scenarios in this guide describe this type of configuration.

However, to increase the availability and scalability of your cache, you can configure multiple instances of OracleAS Web Cache to run as members of a cache cluster. A **cache cluster**, also known as OracleAS Cluster (Web Cache), is a loosely coupled collection of cooperating Web cache instances working together to provide a single logical cache.

This chapter contains the following topics:

- Overview of Cache Clusters
- Benefits of Cache Clusters
- How Cache Clusters Work
- How Cache Content Is Distributed
- Failure Detection and Failover

## Overview of Cache Clusters

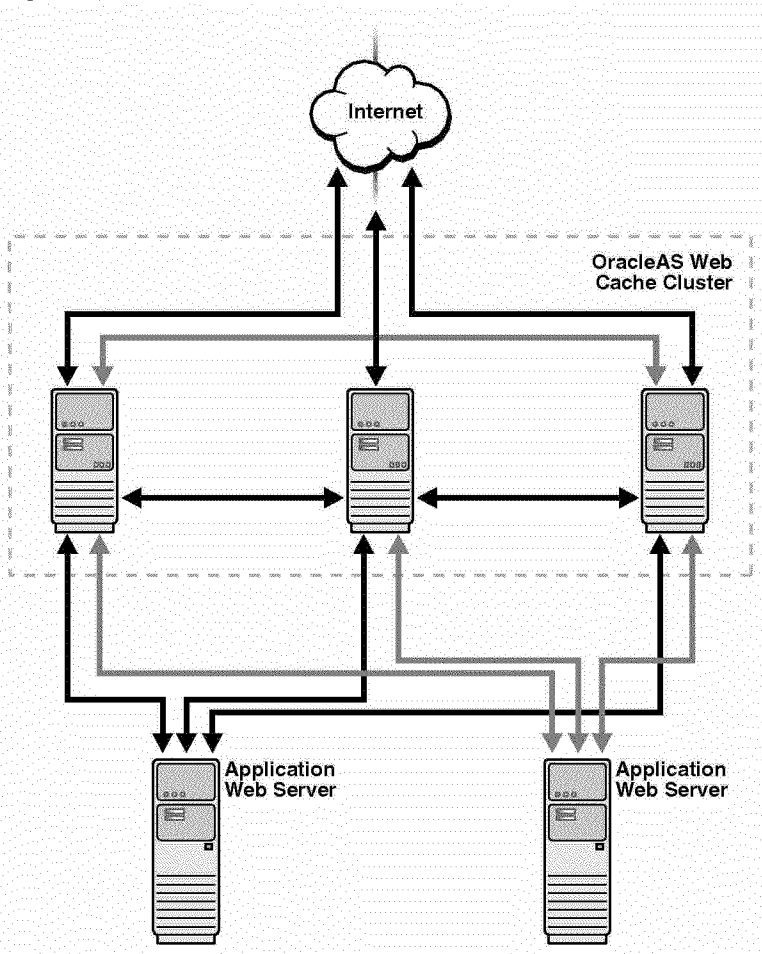
In a cache cluster, multiple instances of OracleAS Web Cache, the **cache cluster member**, operate as one logical cache. A cache cluster can consist of two or more members. The cache cluster members communicate with one another to request cacheable content that is cached by another cache cluster member and to detect when a cache cluster member fails.

Figure 3–1 on page 3-2 shows an OracleAS Web Cache cluster that contains three cache cluster members. As the figure shows, the cluster members communicate with one another as well as with the application Web servers and with the clients.



ORCL00513953

*Figure 3–1 OracleAS Web Cache Cluster Architecture*




OracleAS Web Cache uses the relative capacity of each cache instance to distribute the cached content among the cache cluster members. In effect, it assigns a cache cluster member to be the owner of a particular object. This content is called **owned content**.

In addition to the owned content, OracleAS Web Cache stores popular objects in the cache of each cluster member. These objects are known as **on-demand content**. By storing the on-demand content, OracleAS Web Cache responds to requests for those objects quickly and decreases the number of cache misses. Fewer requests are sent to the application Web server. The result is improved performance.

## Benefits of Cache Clusters

Cache clusters provide the following benefits:

- High availability

  With or without cache clusters, OracleAS Web Cache ensures that cache misses are directed to the most available, highest-performing Web server. With cache clusters, OracleAS Web Cache supports failure detection and failover of caches. If a Web cache fails, other members of the cache cluster detect the failure and take over ownership of the cacheable content of the failed cluster member.

- Scalability and performance



ORCL00513954

By distributing the site's content across multiple caches, more content can be cached and more client connections can be supported, expanding the capacity of your Web site.

By deploying multiples caches in a cache cluster, you make use of the processing power of more CPUs. Because multiple requests are executed in parallel, you increase the number of requests that are served concurrently.

Network bottlenecks often limit the number of requests that can be processed at one time. Even on a node with multiple network cards, you can encounter operating system limitations. By deploying caches on separate nodes, more network bandwidth is available. Response time is improved because of the distribution of requests.

In a cache cluster, fewer requests are routed to the application Web server. Retrieving content from a cache (even if that request is routed to another cache in the cluster) is more efficient than materializing the content from the application Web server.

- Reduced load on the application Web server

  In a cache cluster environment, popular objects are stored in more than one cache. If a cache fails, requested cacheable objects are likely to be stored in the cache of surviving cluster members. As a result, fewer requests for cacheable objects need to be routed to the application Web server even when a cache fails.

  When a failed cache returns to operation, it has no objects cached. In a noncluster environment with multiple independent caches, that cache must route cache misses to the application Web server. In a cache cluster environment, that cache can route cache misses to other caches in the cluster, reducing the load on the application Web server.

  Cache clusters maximize system resource utilization. When each cache in a cache cluster resides on a separate node, more memory is available than for one cache on a single node. With more memory, OracleAS Web Cache can cache more content, resulting in fewer requests to the application Web server.

- Improved data consistency

  Because OracleAS Web Cache uses one set of invalidation rules for all cache cluster members and because it makes it easy to propagate invalidation requests to all cache cluster members, the cached data is more likely to be consistent across all caches in a cluster.

  You can configure a cache cluster that does not support requests between cache cluster members, but allows propagating invalidation requests, as well as propagating configuration changes. See "Configuring Administration and Invalidation-Only Clusters" on page 10-10 for more information.

- Manageability

  Cache clusters are easy to manage because they use one configuration for all cache cluster members. For example, you specify one set of caching rules and one set of invalidation rules. OracleAS Web Cache distributes those rules throughout the cluster by propagating the configuration to each cluster member.

## How Cache Clusters Work

In a cache cluster, multiple instances of OracleAS Web Cache operate as one logical cache.



ORCL00513955

A cache cluster uses one configuration that is propagated to all cluster members. The configuration contains general information, such as security, session information, and caching rules, which is the same for all cluster members. It also contains cache-specific information, such as capacity, administration and other ports, resource limits, and log files, for each cluster member.

Each member must be authenticated before it is added to the cache cluster. The authentication requires that the administration username and password of the OracleAS Web Cache instance to be added be the same as the administration username and password of the cluster.

When you add a cache to the cluster, the cache-specific information of the new cluster member is added to the configuration of the cache cluster. Then, OracleAS Web Cache propagates the configuration to all members of the cluster. Because adding a new member changes the relative capacity of each Web cache, OracleAS Web Cache uses the information about capacity to recalculate which cluster member owns which content.

When cache cluster members detect the failure of another cluster member, the remaining cache cluster members automatically take over ownership of the content of the failing member. When the cache cluster member is reachable again, OracleAS Web Cache again reassigns the ownership of the content.

When you remove a Web cache from a cache cluster, the remaining cache cluster members take over ownership of the content of the removed member. In addition, the configuration information about the removed member is deleted from the configuration and the revised configuration is propagated to the remaining cache cluster members.

In a cache cluster, administrators can decide whether to propagate invalidation messages to all cache cluster members or to send invalidation messages individually to cache cluster members.

> **See Also:** "Invalidation in Cache Clusters" on page 13-5 for more information about invalidation propagation in cache clusters

## How Cache Content Is Distributed

OracleAS Web Cache uses the relative capacity of each cache to automatically distribute ownership of objects among the cache cluster members. For example, in a three-cache cluster, if cache_X has a capacity of 10, cache_Y has a capacity of 10, and cache_Z has a capacity of 20, OracleAS Web Cache distributes ownership of 25% of the cached content to cache_X, 25% of the cached content to cache_Y, and 50% of the cached content to cache_Z.

OracleAS Web Cache maintains a structure to record ownership of objects. When a request for a cacheable object is received, OracleAS Web Cache uses the structure to assign a cache cluster member to be the owner of the object.

Note that in addition to the owned content, OracleAS Web Cache stores popular objects (on-demand content) in the Web cache of each cluster member. By storing the on-demand content, OracleAS Web Cache returns future requests for those objects quickly and decreases the number of cache misses. The result is improved performance.

When an incoming request for a non-cacheable object is received by one of the cache cluster members, the request is forwarded to the application Web server.

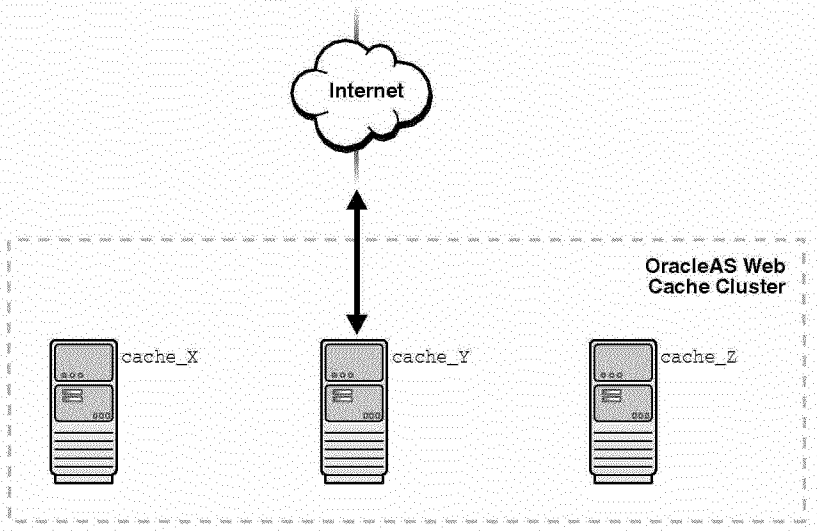
When an incoming request for a cacheable object is received by one of the cache cluster members, what happens next depends on whether or not the requested content is



ORCL00513956

cached by that cluster member and whether or not the content is owned by that cluster member. Suppose that cluster member cache_Y receives a request for cacheable content:

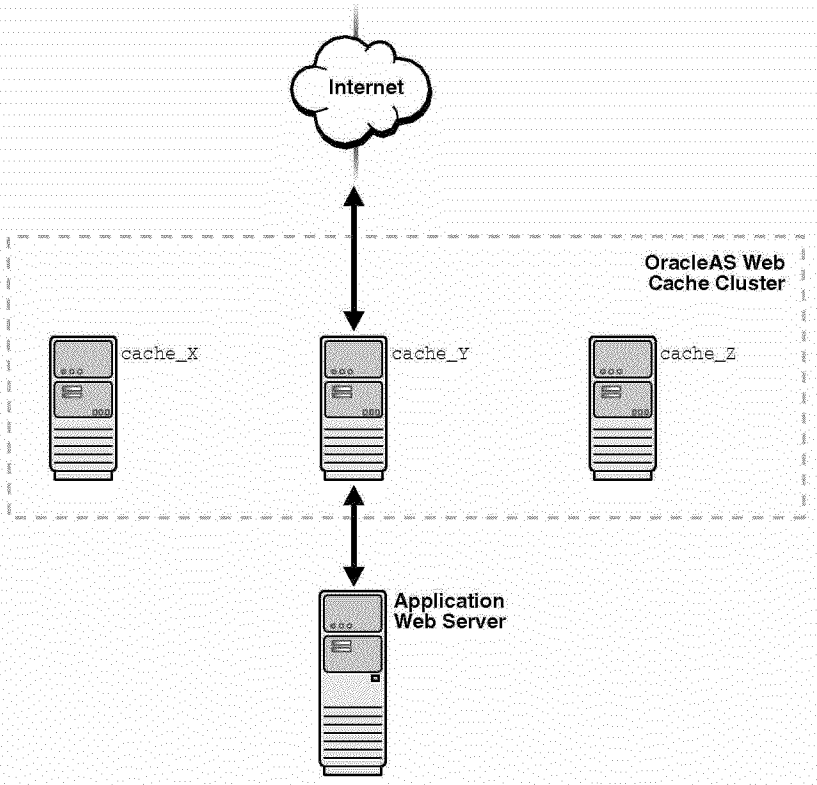
- If the content is cached by cache_Y, cache_Y returns the content to the client, as shown in the following figure. The content could be either owned content or on-demand content.




- If the content is not cached by cache_Y, OracleAS Web Cache performs an ownership lookup. Then:
  - If cache_Y is the owner of the requested content, it sends the request to the application Web server, which returns the requested content to cache_Y. Then, cache_Y caches the content and returns it to the client, as shown in the following figure:

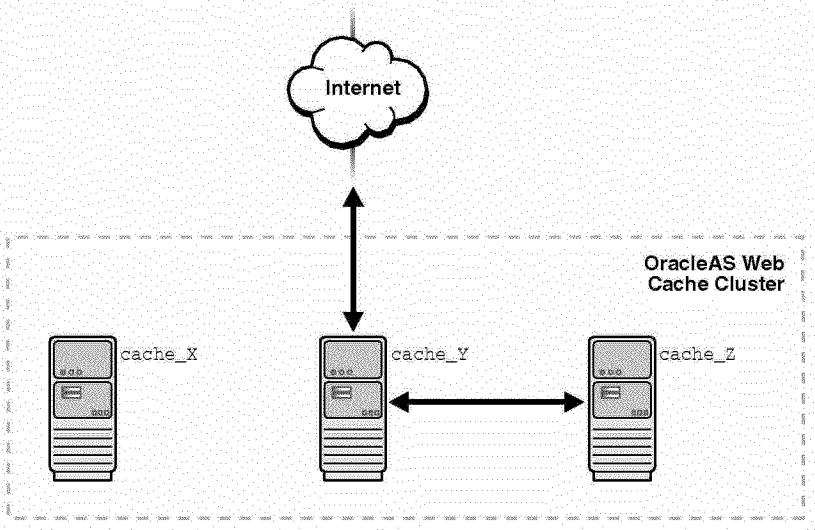


ORCL00513957




– If cache_Y is the not owner of the requested content, cache_Y sends the request to the owner cache, cache_Z. Then:

  * If the requested content is stored in the owner's cache, the owner returns the requested content to the cluster member (cache_Y) that originally received the request. Cache_Y returns the requested content to the client, as shown in the following figure. The content is stored in cache_Y as on-demand content to satisfy future requests for that object quickly.

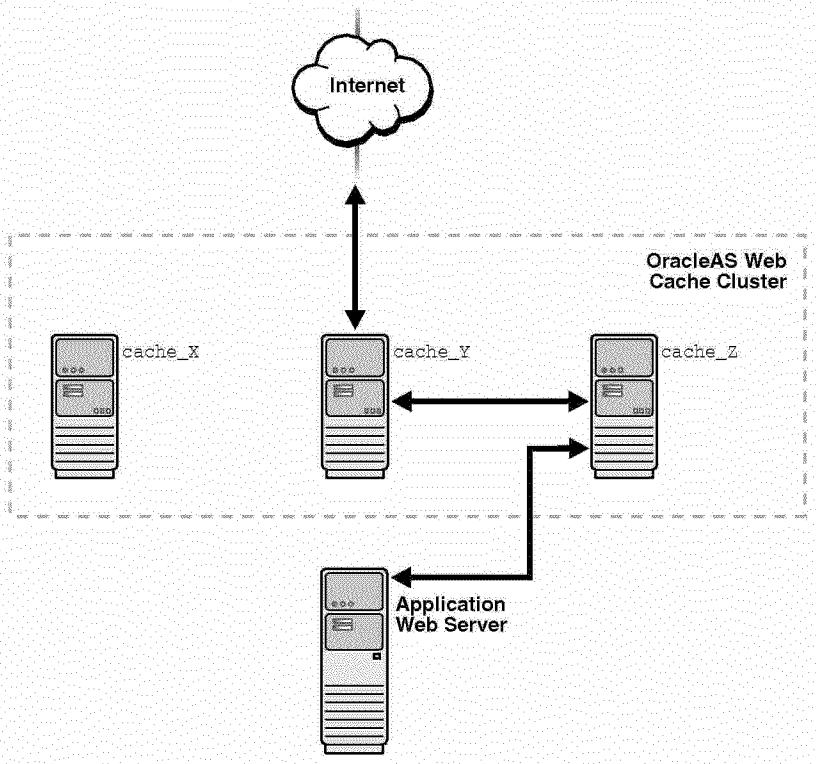



  * If the requested content is not stored in the owner's cache, the owner, cache_Z, sends the request to the application Web server. The application Web server returns the requested content to the owner. The owner caches



ORCL00513958

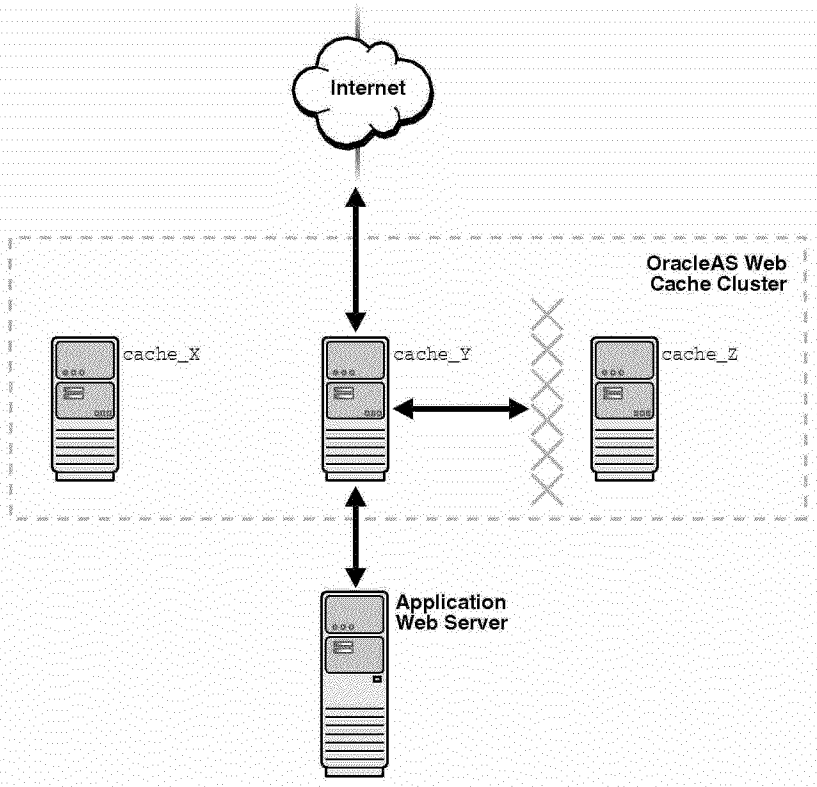
the content and sends the requested content to the cluster member (cache_Y) that originally received the request. Cache_Y returns the requested content to the client and stores the content as on-demand content, as shown in the following figure:




- If cache_Y cannot contact the owner (because of network problems or because the server for the owner has failed), it sends the request to the application Web server. The application Web server returns the requested content to cache_Y. Then, cache_Y returns the requested content to the client and stores the content as on-demand content, as shown in the following figure:



ORCL00513959




When you add a member to or remove a member from the cache cluster, OracleAS Web Cache uses the information about capacity to recalculate which cluster member owns which objects. If the ownership of an object changes and the object is currently cached, OracleAS Web Cache designates the object as on-demand content, rather than owned content, for that cache. The object is not removed from the previous owner cache, nor is it moved to the new owner cache. The object is not cached in the new owner cache until another request for the object is received.

## Failure Detection and Failover

OracleAS Web Cache clusters ensure high availability through failure detection and failover. In clusters, **failure detection** ensures that OracleAS Web Cache can detect when a cache cluster member is unavailable; **failover** ensures that OracleAS Web Cache transfers ownership of the content of the failing member to the remaining cluster members.

Cache cluster members send requests to the cluster member who is the owner of the requested content. If a cache cluster member does not receive a response from another cluster member after a specified failover threshold (the number of consecutive attempts to reach a cache), the cache cluster member assumes that the other cluster member has failed. Each cluster member individually detects the failure of other cluster members.

As each cache cluster member detects the failure of another cluster member, it recalculates the relative capacity of the remaining cache cluster members. Then, it reassigns ownership of objects based on the new relative capacity and the ownership array. Note that although ownership is reassigned, the content is not cached in the new owner cache until another request for the object is received.



ORCL00513960

The cache cluster members poll the failed Web cache server for its current status until it is reachable again. When the failed Web cache is reachable, it rejoins the cache cluster. Each cache cluster member again recalculates the relative capacity of the cache cluster members and reassigns ownership of the objects.

**See Also:** Chapter 10 for information about configuring a cache cluster, including specifying a failover threshold



ORCL00513961

ORCL00513962

# Exhibit 12

# Oracle® XML DB

Developer's Guide

10*g* Release 2 (10.2)

**B14259-02**

August 2005

This manual describes Oracle XML DB. It includes guidelines and examples for loading, storing, generating, manipulating, managing, accessing, and querying XML data in Oracle Database.

ORACLE

Oracle XML DB Developer's Guide, 10*g* Release 2 (10.2)

B14259-02

Copyright © 2002, 2005, Oracle. All rights reserved.

Primary Author: Drew Adams

Contributing Author: Nipun Agarwal, Abhay Agrawal, Omar Alonso, David Anniss, Sandeepan Banerjee, Mark Bauer, Ravinder Booreddy, Stephen Buxton, Yuen Chan, Sivasankaran Chandrasekar, Vincent Chao, Ravindranath Chennoju, Dan Chiba, Mark Drake, Fei Ge, Wenyun He, Shelley Higgins, Thuvan Hoang, Sam Idicula, Namit Jain, Neema Jalali, Bhushan Khaladkar, Viswanathan Krishnamurthy, Muralidhar Krishnaprasad, Geoff Lee, Wesley Lin, Annie Liu, Anand Manikutty, Jack Melnick, Nicolas Montoya, Steve Muench, Ravi Murthy, Eric Paapanen, Syam Pannala, John Russell, Eric Sedlar, Vipul Shah, Cathy Shea, Asha Tarachandani, Tarvinder Singh, Simon Slack, Muralidhar Subramanian, Asha Tarachandani, Priya Vennapusa, James Warner

Contributor: Reema Al-Shaikh, Harish Akali, Vikas Arora, Deanna Bradshaw, Paul Brandenstein, Lisa Eldridge, Craig Foch, Wei Hu, Reema Koo, Susan Kotsovolos, Sonia Kumar, Roza Leyderman, Zhen Hua Liu, Diana Lorentz, Yasuhiro Matsuda, Valarie Moore, Bhagat Nainani, Visar Nimani, Sunitha Patel, Denis Raphaely, Rebecca Reitmeyer, Ronen Wolf

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software—Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

# 1

# Introduction to Oracle XML DB

This chapter introduces the features and architecture of Oracle XML DB. It contains these topics:

- Features of Oracle XML DB
- Oracle XML DB Architecture
- Oracle XML DB Features
- Oracle XML DB Benefits
- Searching XML Data Stored in CLOBs Using Oracle Text
- Building Messaging Applications using Oracle Streams Advanced Queuing
- Requirements for Running Oracle XML DB
- Standards Supported by Oracle XML DB
- Oracle XML DB Technical Support
- Oracle XML DB Examples Used in This Manual
- Further Oracle XML DB Case Studies and Demonstrations

## Features of Oracle XML DB

Oracle XML DB is the name for a set of Oracle Database technologies related to high-performance XML storage and retrieval. It provides native XML support by encompassing both SQL and XML data models in an interoperable manner.

Oracle XML DB includes the following features:

- Support for the World Wide Web Consortium (W3C) XML and XML Schema data models and standard access methods for navigating and querying XML. The data models are incorporated into Oracle Database.
- Ways to store, query, update, and transform XML data while accessing it using SQL.
- Ways to perform XML operations on SQL data.
- A simple, lightweight XML repository where you can organize and manage database content, including XML, using a file/folder/URL metaphor.
- A storage-independent, content-independent and programming language-independent infrastructure for storing and managing XML data. This provides new ways of navigating and querying XML content stored in the database. For example, Oracle XML DB Repository facilitates this by managing XML document hierarchies.

- Industry-standard ways to access and update XML. The standards include the W3C XPath recommendation and the ISO-ANSI SQL/XML standard. FTP, HTTP(S), and WebDAV can be used to move XML content into and out of Oracle Database. Industry-standard APIs provide programmatic access and manipulation of XML content using Java, C, and PL/SQL.
- XML-specific memory management and optimizations.
- Enterprise-level Oracle Database features for XML content: reliability, availability, scalability, and security.

Oracle XML DB can be used in conjunction with Oracle XML Developer's Kit (XDK) to build applications that run in the middle tier in either Oracle Application Server or Oracle Database.

**See Also:** *Oracle XML Developer's Kit Programmer's Guide*

## Oracle XML DB Architecture

Figure 1–1 and Figure 1–2 show the software architecture of Oracle XML DB. The two main features are:

- Storage of `XMLType` tables and views
- Oracle XML DB Repository

*Figure 1–1 Oracle XML DB Architecture: XMLType Storage and Repository*

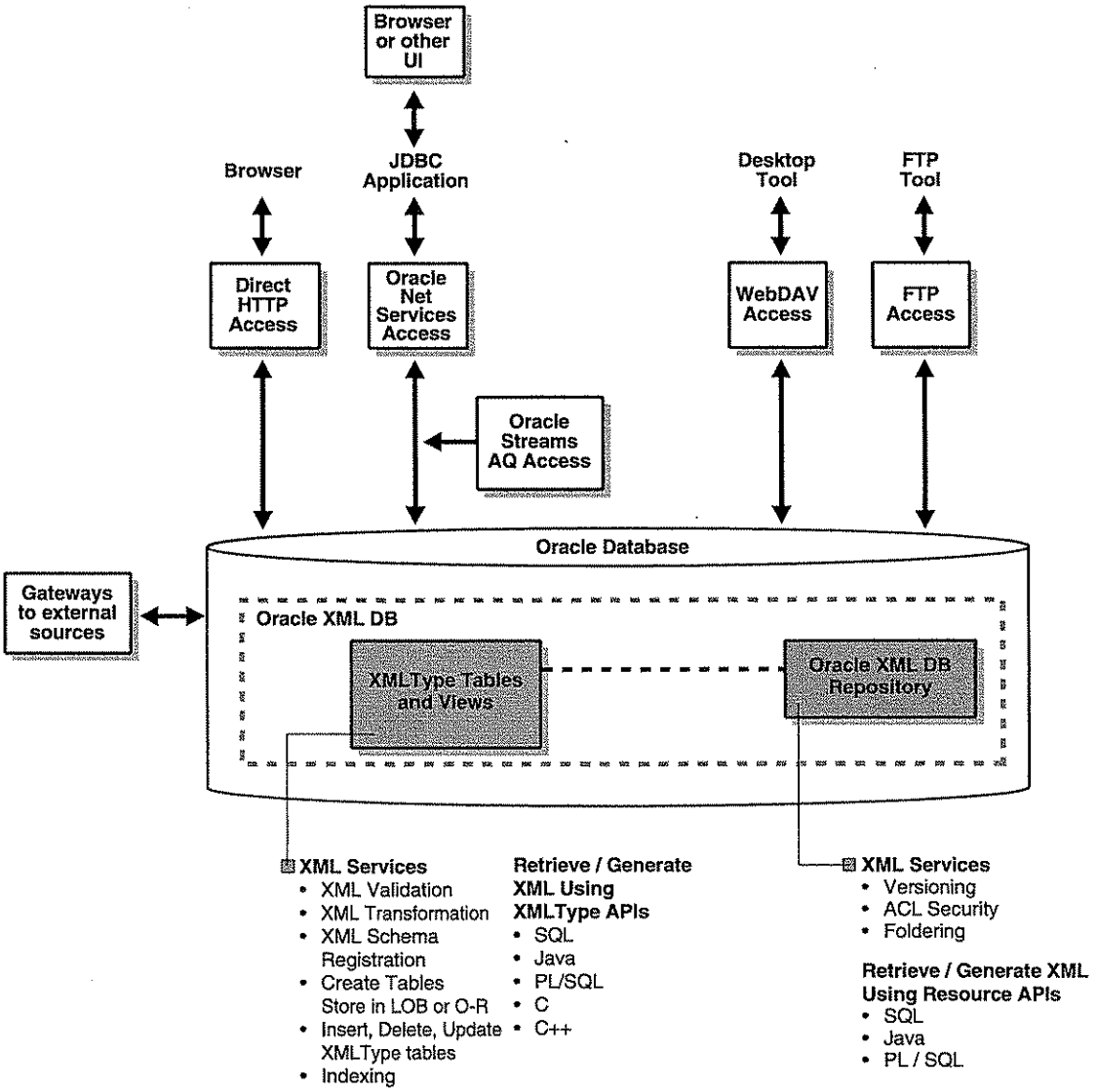
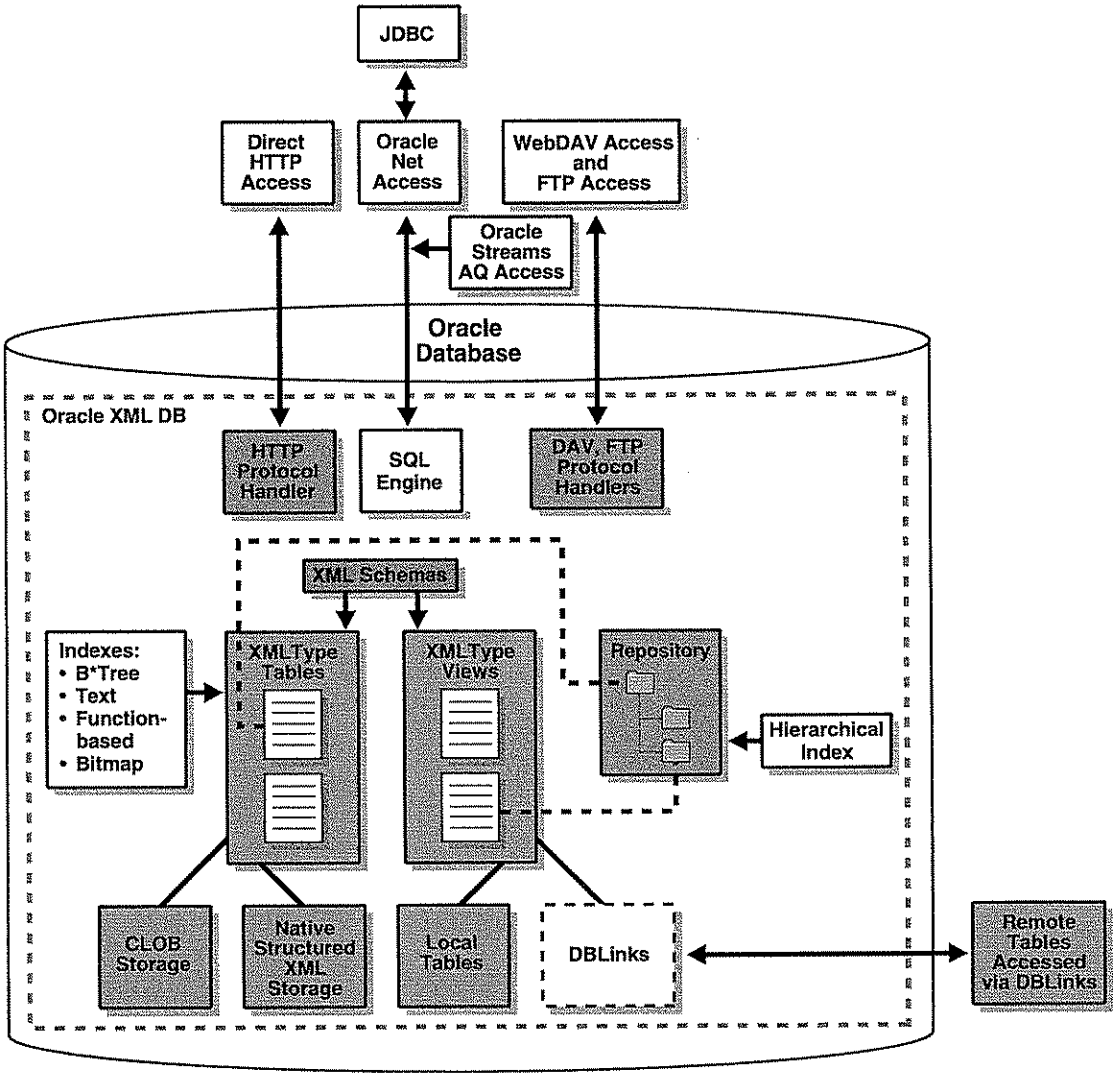


*Figure 1–2 Oracle XML DB Architecture: XMLType Storage*




## XMLType Storage

Figure 1–2 shows `XMLType` storage in Oracle XML DB.

When XML schemas are registered with Oracle XML DB, a set of default tables are created and used to store XML instance documents associated with the schema. These documents can be viewed and accessed in Oracle XML DB Repository.

`XMLType` tables and columns can be stored as Character Large Object (`CLOB`) values or as a set of objects. When stored as a set of objects, we refer to **structured**, or **shredded** storage.

natively, using structured XML, or in Character Large Object (`CLOB`) values.

Data in `XMLType` views can be stored in local or remote tables. Remote tables can be accessed through database links.

Both `XMLType` tables and views can be indexed using B*Tree, Oracle Text, function-based indexes, or bitmap indexes.

You can access data in Oracle XML DB Repository using any of the following:

- HTTP(S), through the HTTP protocol handler.
- WebDAV and FTP, through the WebDAV and FTP protocol server.
- SQL, through Oracle Net Services, including Java Database Connectivity (JDBC).

Oracle XML DB supports XML data messaging using Oracle Streams Advanced Queuing (AQ) and Web Services.

**See Also:**

- Part II. "Storing and Retrieving XML Data in Oracle XML DB"
- Chapter 25, "FTP, HTTP(S), and WebDAV Access to Repository Data"
- Chapter 31, "Exchanging XML Data with Oracle Streams AQ"

## APIs for XML

Table 1–1 lists the reference documentation for the PL/SQL, C, and C++ Application Programming Interfaces (APIs) that you can use to manipulate XML documents and data. The main reference for PL/SQL, C, and C++ APIs is *Oracle Database PL/SQL Packages and Types Reference.*

**See Also:** *Oracle Database XML Java API Reference* for information on Java APIs for XML

***Table 1–1 APIs Related to XML***

| API | Documentation | Description |
|---|---|---|
| XMLType | *Oracle Database PL/SQL Packages and Types Reference*, Chapter "XMLType", *Oracle Database XML C API Reference*, and *Oracle Database XML C++ API Reference* | PL/SQL, C, and C++ APIs with XML operations on XMLType data – validation, transformation. |
| Database URI types | *Oracle Database PL/SQL Packages and Types Reference*, Chapter "Database URI TYPEs" | Functions used for various URI types. |
| DBMS_XDB | *Oracle Database PL/SQL Packages and Types Reference*, Chapter "DBMS_XDB" | PL/SQL API for managing Oracle XML DB Repository resources, ACL-based security, and configuration sessions. |
| DBMS_XDB_VERSION | *Oracle Database PL/SQL Packages and Types Reference*, Chapter "DBMS_XDB_VERSION" | PL/SQL API for version management of repository resources. |
| DBMS_XDBT | *Oracle Database PL/SQL Packages and Types Reference*, Chapter "DBMS_XDBT" | PL/SQL API for creation of text indexes onrepository resources. |
| DBMS_XDBZ | *Oracle Database PL/SQL Packages and Types Reference*, Chapter "DBMS_XDBZ" | Oracle XML DB Repository ACL-based security. |
| DBMS_XMLDOM | *Oracle Database PL/SQL Packages and Types Reference*, Chapter "DBMS_XMLDOM" | PL/SQL implementation of the DOM API for XMLType. |
| DBMS_XMLGEN | *Oracle Database PL/SQL Packages and Types Reference*, Chapter "DBMS_XMLGEN" | PL/SQL API for transformation of SQL query results into canonical XML format. |

*Table 1–1 (Cont.) APIs Related to XML*

| API | Documentation | Description |
|---|---|---|
| DBMS_XMLPARSER | *Oracle Database PL/SQL Packages and Types Reference*, Chapter "DBMS_XMLPARSER" | PL/SQL implementation of the DOM Parser API for XMLType. |
| DBMS_XMLQUERY | *Oracle Database PL/SQL Packages and Types Reference*, Chapter "DBMS_XMLQUERY" | PL/SQL API providing database-to-XMLType functionality. (Where possible, use DBMS_XMLGEN instead.) |
| DBMS_XMLSAVE | *Oracle Database PL/SQL Packages and Types Reference*, Chapter "DBMS_XMLSAVE" | PL/SQL API providing XML- to-database type functionality. |
| DBMS_XMLSCHEMA | *Oracle Database PL/SQL Packages and Types Reference*, Chapter "DBMS_XMLSCHEMA" | PL/SQL API for managing XML schemas within Oracle Database – schema registration, deletion. |
| DBMS_XMLSTORE | *Oracle Database PL/SQL Packages and Types Reference*, Chapter "DBMS_XMLSTORE" | PL/SQL API for storing XML data in relational tables. |
| DBMS_XSLPROCESSOR | *Oracle Database PL/SQL Packages and Types Reference*, Chapter "DBMS_XSLPROCESSOR" | PL/SQL implementation of an XSLT processor. |

## XML Schema Catalog Views

Table 1–2 lists the XML schema catalog views for Oracle XML DB, which provide access to metadata about XML schemas that are registered with Oracle XML DB. Information about a given view can be obtained by using the SQL command DESCRIBE. Example:

```
DESCRIBE USER_XML_SCHEMAS
```

*Table 1–2 XML Schema Catalog Views*

| Schema | Description |
|---|---|
| USER_XML_SCHEMAS | Registered XML schemas *owned* by the current *user* |
| ALL_XML_SCHEMAS | Registered XML schemas *usable* by the current *user* |
| DBA_XML_SCHEMAS | Registered XML schemas in Oracle XML DB |
| USER_XML_TABLES | XMLType tables *owned* by the current *user* |
| ALL_XML_TABLES | XMLType tables *usable* by the current *user* |
| DBA_XML_TABLES | XMLType tables in Oracle XML DB |
| USER_XML_TAB_COLS | XMLType table columns *owned* by the current *user* |
| ALL_XML_TAB_COLS | XMLType table columns *usable* by the current *user* |
| DBA_XML_TAB_COLS | XMLType table columns in Oracle XML DB |
| USER_XML_VIEWS | XMLType views *owned* by the current *user* |
| ALL_XML_VIEWS | XMLType views *usable* by the current *user* |
| DBA_XML_VIEWS | XMLType views in Oracle XML DB |
| USER_XML_VIEW_COLS | XMLType view columns *owned* by the current *user* |
| ALL_XML_VIEW_COLS | XMLType view columns *usable* by the current *user* |
| DBA_XML_VIEW_COLS | XMLType view columns in Oracle XML DB |

**See Also:** *Oracle Database PL/SQL Packages and Types Reference*

## Views RESOURCE_VIEW and PATH_VIEW

Oracle XML DB views `RESOURCE_VIEW` and `PATH_VIEW` provide SQL access to data in Oracle XML DB Repository through protocols such as FTP and WebDAV. View `PATH_VIEW` has one row for each unique path in the repository; view `RESOURCE_VIEW` has one row for each resource in the repository.

The Oracle XML DB resource API for PL/SQL, `DBMS_XDB`, provides query and DML functions. It is based on `RESOURCE_VIEW` and `PATH_VIEW`.

**See Also:** Chapter 22, "SQL Access Using RESOURCE_VIEW and PATH_VIEW"

## Overview of Oracle XML DB Repository

Oracle XML DB Repository is a component of Oracle Database that is optimized for handling XML data. The Oracle XML DB repository contains **resources**, which can be either **folders** (directories, containers) or files. Each resource has these properties:

- It is identified by a *path* and *name*.
- It has *content* (data), which can be XML data but need not be.
- It has a set of **system-defined metadata** (properties), such as `Owner` and `CreationDate`, in addition to its content. Oracle XML DB uses this information to manage the resource.
- It might also have **user-defined metadata**: information that is not part of the content, but is associated with it.
- It has an associated **access control list** that determines who can access the resource, and for what operations.

Although Oracle XML DB Repository treats XML content specially, you can use Oracle XML DB Repository to store other kinds of data, besides XML; in fact, you can use the repository to access *any* data that is stored in Oracle Database.

**See Also:**

- Part V. "Oracle XML DB Repository: Foldering, Security, and Protocols"
- Chapter 25, "FTP, HTTP(S), and WebDAV Access to Repository Data" for information on accessing XML data in `XMLType` tables and columns using external protocols
- Chapter 26, "User-Defined Repository Metadata"

### Accessing and Manipulating XML in the Oracle XML DB Repository

You can access data in Oracle XML DB Repository in the following ways (see Figure 1–1):

- Using SQL, through views `RESOURCE_VIEW` and `PATH_VIEW`
- Using PL/SQL, through the `DBMS_XDB` API
- Using Java, through the Oracle XML DB resource API for Java

### XML Services

Besides supporting APIs that access and manipulate data, Oracle XML DB Repository provides APIs for the following services:

- **Versioning.** Oracle XML DB uses the `DBMS_XDB_VERSION` PL/SQL package for versioning resources in Oracle XML DB Repository. Subsequent updates to a resource create a new version (the data corresponding to previous versions is retained). Versioning support is based on the IETF WebDAV standard.
- **ACL Security.** Oracle XML DB resource security is based on Access Control Lists (ACLs). Every resource in Oracle XML DB has an associated ACL that lists its privileges. Whenever resources are accessed or manipulated, the ACLs determine if the operation is legal. An ACL is an XML document that contains a set of Access Control Entries (ACE). Each ACE grants or revokes a set of permissions to a particular user or group (database role). This access control mechanism is based on the WebDAV specification.
- **Foldering.** Oracle XML DB Repository manages a persistent hierarchy of folder (directory) resources that contain other resources (files or folders). Oracle XML DB modules, such as protocol servers, the schema manager, and the Oracle XML DB `RESOURCE_VIEW` API, use foldering to map path names to resources.

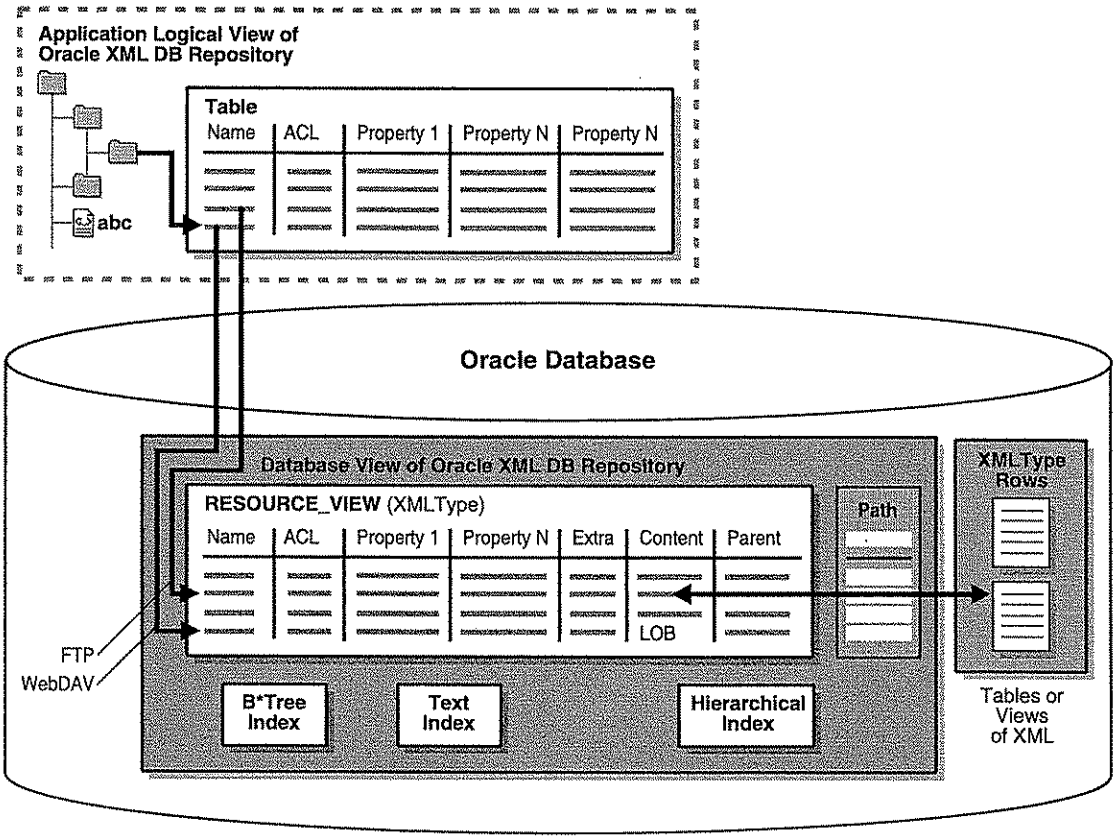
### Oracle XML DB Repository Architecture

Figure 1–3 describes the Oracle XML DB Repository architecture. You can access the repository in SQL, for example, using the `RESOURCE_VIEW` API. In addition to the resource information, the `RESOURCE_VIEW` also contains a `Path` column, which holds the paths to each resource.

**See Also:**

- Chapter 20, "Accessing Oracle XML DB Repository Data"
- Chapter 22, "SQL Access Using RESOURCE_VIEW and PATH_VIEW"

*Figure 1–3 Oracle XML DB Repository Architecture*



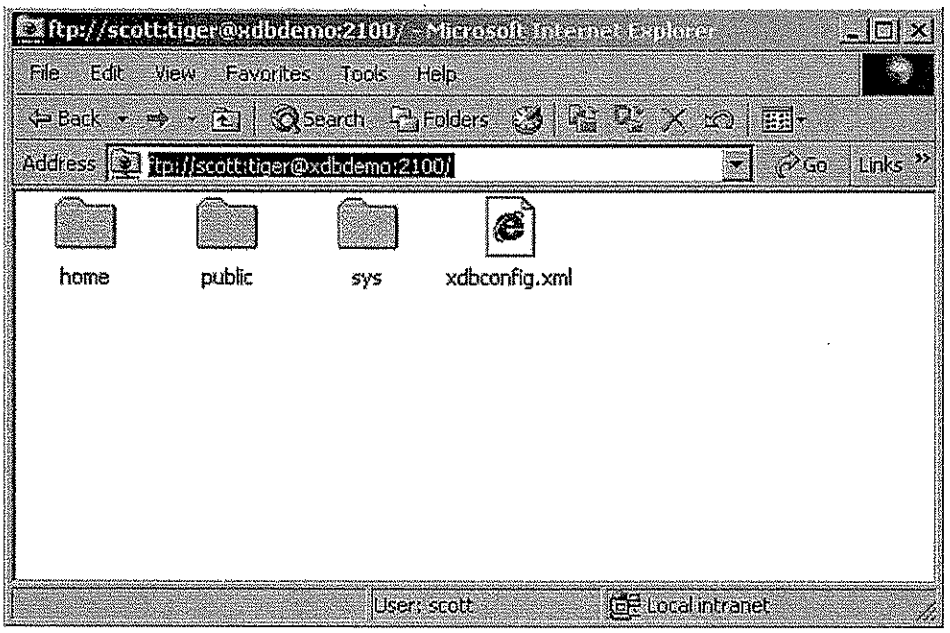

## How Does Oracle XML DB Repository Work?

The relational model table-row-column metaphor, is accepted as an effective mechanism for managing structured data. The model is not as effective for managing semistructured and unstructured data, such as document- or content-oriented XML. For example, a book is not easily represented as a set of rows in a table. It is more natural to represent a book as a hierarchy, book:chapter:section:paragraph, and to represent the hierarchy as a set of folders and subfolders.

- A hierarchical metaphor manages document-centric XML content. Relational databases are traditionally poor at managing hierarchical structures and traversing a path or URL. Oracle XML DB provides a hierarchically organized repository that can be queried and through which document-centric XML content can be managed.
- A hierarchical index speeds up folder and path traversals. Oracle XML DB includes a new, patented hierarchical index that speeds up folder and path traversals in Oracle XML DB Repository. The hierarchical index is transparent to end users, and allows Oracle XML DB to perform folder and path traversals at speeds comparable to or faster than conventional file systems.
- You can access XML documents in Oracle XML DB Repository using standard connect-access protocols such as FTP, HTTP(S), and WebDAV, in addition to languages SQL, PL/SQL, Java, and C. The repository provides content authors and editors direct access to XML content stored in Oracle Database.

- A resource in this context is a file or folder, identified by a URL. WebDAV is an IETF standard that defines a set of extensions to the HTTP protocol. It allows an HTTP server to act as a file server for a DAV-enabled client. For example, a WebDAV-enabled editor can interact with an HTTP/WebDAV server as if it were a file system. The WebDAV standard uses the term **resource** to describe a file or a folder. Each resource managed by a WebDAV server is identified by a URL. Oracle XML DB adds native support to Oracle Database for these protocols. The protocols were designed for document-centric operations. By providing support for these protocols, Oracle XML DB allows Windows Explorer, Microsoft Office, and products from vendors such as Altova, Macromedia, and Adobe to work directly with XML content stored in Oracle XML DB Repository. Figure 1–4 shows the root-level directory of the repository as seen from Microsoft Web Folder.

***Figure 1–4 Microsoft Web Folder View of Oracle XML DB Repository***




**See Also:** Chapter 3, "Using Oracle XML DB"

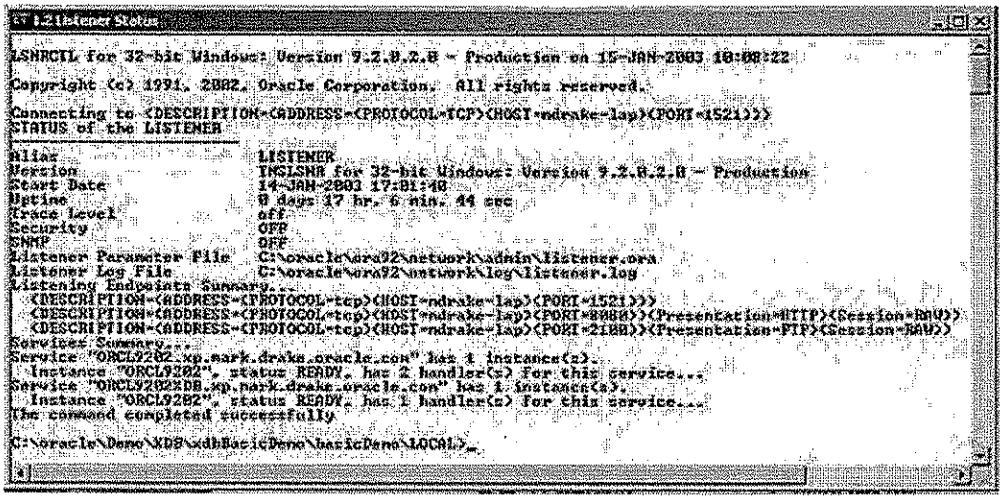
Hence, WebDAV clients such as Microsoft Windows Explorer can connect directly to Oracle XML DB Repository. No additional Oracle Database or Microsoft-specific software or other complex middleware is needed. End users can work directly with Oracle XML DB Repository using familiar tools and interfaces.

### Oracle XML DB Protocol Architecture

One key feature of the Oracle XML DB architecture is that HTTP(S), WebDAV, and FTP protocols are supported using the same architecture used to support Oracle Data Provider for .NET (ODP.NET) in a shared server configuration. The Listener listens for HTTP(S) and FTP requests in the same way that it listens for ODP .NET service requests. When the Listener receives an HTTP(S) or FTP request, it hands it off to an Oracle Database shared server process which services it and sends the appropriate response back to the client.

You can use the TNS Listener command `lsnrctl status` to verify that HTTP(S) and FTP support has been enabled – see Figure 1–5.

*Figure 1–5 Listener Status with FTP and HTTP(S) Protocol Support Enabled*




**See Also:** Chapter 25, "FTP, HTTP(S), and WebDAV Access to Repository Data"

## Programmatic Access to Oracle XML DB (Java, PL/SQL, and C)

All Oracle XML DB functionality is accessible from C, PL/SQL, and Java. The most popular ways to build web-based applications are these:

- Using servlets and JSPs (Java Server Pages). A typical API accesses data using Java Database Connectivity (JDBC).
- Using XSL and XSPs (XML Style Sheets plus XML Server Pages). A typical API accesses data in the form of XML documents that are processed using a Document Object Model (DOM) API implementation.

Oracle XML DB supports both of these styles of application development. It provides Java, PL/SQL, and C implementations of the DOM API.

Applications that use JDBC, such as those based on servlets, need prior knowledge of the data structure they are processing. Oracle JDBC drivers allow you to access and update `XMLType` tables and columns, and call PL/SQL procedures that access Oracle XML DB Repository.

Applications that use DOM, such as those based on XSLT transformations, typically require less knowledge of the data structure. DOM-based applications use string names to identify pieces of content, and must dynamically walk through the DOM tree to find the required information. For this Oracle XML DB supports the use of the DOM API to access and update `XMLType` columns and tables. Programming to a DOM API is more flexible than programming through JDBC, but it may require more resources at run time.

# Oracle XML DB Features

Any database used for managing XML must be able to persist XML documents. Oracle XML DB is capable of much more than this. It provides standard database features such as transaction control, data integrity, replication, reliability, availability, security, and scalability., while also allowing for efficient indexing, querying, updating, and searching of XML documents in an XML-centric manner.

The hierarchical nature of XML presents the traditional relational database with a number of challenges:

- In a relational database the *table-row* metaphor locates content. Primary-Key Foreign-Key relationships help define the relationships between content. Content is accessed and updated using the table-row-column metaphor.
- XML on the other hand uses *hierarchical* techniques to achieve the same functionality. A URL is used to locate an XML document. URL-based standards such as XLink are used to defined the relationships between XML documents. W3C Recommendations like XPath are used to access and update content contained within XML documents. Both URLs and XPath expressions are based on *hierarchical* metaphors. A URL uses a path through a *folder hierarchy* to identify a document, whereas XPath uses a path through the *node hierarchy* of an XML document to access part of an XML document.

Oracle XML DB addresses these challenges by introducing new SQL functions and methods that allow the use of XML-centric metaphors, such as XPath expressions for querying and updating XML Documents.

The major features of Oracle XML DB are these:

- XMLType Datatype
- XML Schema Support
- Structured Versus Unstructured Storage
- XML/SQL Duality
- SQL/XML INCITS Standard SQL Functions
- Rewriting of XPath Expressions: XPath Rewrite
- XMLType Storage. This was described previously on page 1-4.
- Overview of Oracle XML DB Repository. This was described previously on page 1-7.

## XMLType Datatype

`XMLType` is a native server datatype that lets the database understand that a column or table contains XML. This is similar to the way that date and timestamp datatypes let the database understand that a column contains a date. Datatype `XMLType` also provides methods that allow common operations such as XML schema validation and XSL transformations on XML content.

You can use `XMLType` like any other datatype. For example, you can use `XMLType` when:

- Creating a column in a relational table
- Declaring PL/SQL variables
- Defining and calling PL/SQL procedures and functions

Since `XMLType` is an object type, you can also create a *table* of `XMLType`. By default, an `XMLType` table or column can contain any well-formed XML document.

The following example shows creating a simple table with an `XMLType` column.

### Oracle XML DB Stores XML Text in CLOBs

Oracle XML DB stores the content of the document as XML text using the Character Large Object (`CLOB`) datatype. This allows for maximum flexibility in terms of the

shape of the XML structures that can be stored in a single table or column and the highest rates of ingestion and retrieval.

#### XMLType Tables and Columns Can Conform to an XML Schema

`XMLType` tables or columns can be constrained and conform to an XML schema. This has several advantages:

- The database will ensure that only XML documents that validate against the XML schema can be stored in the column or table.
- Since the contents of the table or column conform to a known XML structure, Oracle XML DB can use the information contained in the XML schema to provide more intelligent query and update processing of the XML.
- Constraining the `XMLType` to an XML schema provides the option of storing the content of the document using *structured-storage* techniques. Structured-storage decomposes or **shreds** the content of the XML document and stores it as a set of SQL objects rather than simply storing the document as text in a `CLOB`. The object-model used to store the document is automatically derived from the contents of the XML schema.

#### The XMLType API

Datatype `XMLType` provides the following:

- **Constructors**. These allow an `XMLType` value to be created from a `VARCHAR`, `CLOB`, `BLOB`, or `BFILE` value.
- **Methods**. A number of XML-specific methods that operate on `XMLType` instances. `XMLType` methods provide support for the following common operations:
  - Extract a subset of nodes contained in the `XMLType` – method `extract()`.
  - Check whether or not a particular node exists in the `XMLType` – method `existsNode()`.
  - Validate the contents of the `XMLType` against an XML schema – method `schemaValidate()`.
  - Perform an XSL Transformation – method `transform()`.

  **See Also:** Chapter 4, "XMLType Operations" and Chapter 9, "Transforming and Validating XMLType Data"

## XML Schema Support

Support for the Worldwide Web Consortium (W3C) XML Schema Recommendation is a key feature in Oracle XML DB. XML Schema specifies the structure, content, and certain semantics of a set of XML documents. It is described in detail at `http://www.w3.org/TR/xmlschema-0/`.

#### XML Schema Unifies Document and Data Modeling

XML Schema unifies both *document* and *data* modeling. In Oracle XML DB, you can create tables and types automatically using XML Schema. In short, this means that you can develop and use a standard data model for *all* your data, structured, unstructured, and semistructured. You can use Oracle XML DB to enforce this data model for all your data.

### You Can Create XMLType Tables and Columns, Ensure DOM Fidelity

You can create XML schema-based `XMLType` tables and columns and optionally specify, for example, that they:

- Conform to pre-registered XML schemas
- Are stored in structured storage format specified by the XML schema, maintaining DOM fidelity

### Use XMLType Views to Wrap Relational Data

You can also choose to wrap existing relational and object-relational data into XML format using `XMLType` views.

You can store an `XMLType` object as an XML object that is based on an XML schema or not based on an XML schema:

- *schema-based objects*. These are stored in Oracle XML DB as Large Objects (LOBs) or in structured storage (object-relationally) in tables, columns, or views.
- *Non-schema-based objects*. These are stored in Oracle XML DB as LOBs.

You can map from XML instances to structured or LOB storage. The mapping can be specified in an XML schema, and the schema must be registered in Oracle XML DB. This is a required step before storing XML schema-based instance documents. Once registered, the XML schema can be referenced using its URL.

### W3C Schema for Schemas

The W3C Schema Working Group publishes an XML schema, often referred to as the "Schema for Schemas". This XML schema provides the definition, or vocabulary, of the XML Schema language. An XML schema definition (XSD) is an XML document, that is compliant with the vocabulary defined by the "Schema for Schemas". An XML schema uses vocabulary defined by W3C XML Schema Working Group to create a collection of type definitions and element declarations that declare a shared vocabulary for describing the contents and structure of a new class of XML documents.

### XML Schema Base Set of Data Types Can be Extended

The XML Schema language provides strong typing of elements and attributes. It defines 47 scalar data types. The base set of data types can be extended using object-oriented techniques like inheritance and extension to define more complex types. W3C XML Schema vocabulary also includes constructs that allow the definition of complex types, substitution groups, repeating sets, nesting, ordering, and so on. Oracle XML DB supports all of constructs defined by the XML Schema Recommendation, except for redefines.

XML schemas are most commonly used as a mechanism for checking whether instance documents conform with their specifications (validation). Oracle XML DB includes methods and SQL functions that allow an XML schema to be used for this.

---

**Note:** This manual uses the term **XML schema** (lower-case "s") to reference any XML schema that conforms to the W3C XML Schema (upper-case "S") Recommendation. Since an XML schema is used to define a class of XML documents, the term **instance document** is often used to describe an XML document that conforms to a particular XML schema.

---

**See Also:** Appendix A, "XML Schema Primer" and Chapter 5, "XML Schema Storage and Query: Basic" for more information about using XML schemas with Oracle XML DB

## Structured Versus Unstructured Storage

One key decision to make when using Oracle XML DB for persisting XML documents is when to use *structured storage* and when to use *unstructured* storage.

- **Unstructured storage** provides for the highest possible throughput when inserting and retrieving entire XML documents. It also provides the greatest degree of flexibility in terms of the structure of the XML that can be stored in a `XMLType` table or column. These throughput and flexibility benefits come at the expense of certain aspects of intelligent processing. There is little the database can do to optimize queries or updates on XML stored using a `CLOB` datatype.
- **Structured storage** has a number of advantages for managing XML, including optimized memory management, reduced storage requirements, b-tree indexing and in-place updates. These advantages are at a cost of somewhat increased processing overhead during ingestion and retrieval and reduced flexibility in terms of the structure of the XML that can be managed by a given `XMLType` table or column.

Table 1–3 outlines the merits of structured and unstructured storage.

***Table 1–3 XML Storage Options: Structured or Unstructured***

| | Unstructured Storage | Structured Storage |
|---|---|---|
| Throughput | Highest possible throughput when ingesting and retrieving the entire content of an XML document. | The decomposition process results in slightly reduced throughput when ingesting retrieving the entire content of an XML document. |
| Flexibility | Provides the maximum amount of flexibility in terms of the structure of the XML documents that can be stored in an `XMLType` column or table. | Limited Flexibility. Only documents that conform to the XML schema can be stored in the `XMLType` table or column. Changes to the XML schema may require data to be unloaded and re-loaded. |
| XML Fidelity | Delivers Document Fidelity: Maintains the original XML byte for byte, which may be important to some applications. | *DOM Fidelity:* A DOM created from an XML document that has been stored in the database will be identical to a DOM created from the original document. However trailing new lines, white space characters between tags and some data formatting may be lost. |
| Update Operations | When any part of the document is updated the entire document must be written back to disk. | The majority of update operations can be performed using XPath rewrite. This allows in-place, piece-wise update, leading to significantly reduced response times and greater throughput. |
| XPath based queries | XPath operations evaluated by constructing DOM from `CLOB` and using functional evaluations. This can be very expensive when performing operations on large collections of documents. | XPath operations may be evaluated using XPath rewrite, leading to significantly improved performance, particularly with large collections of documents. |

*Table 1–3 (Cont.) XML Storage Options: Structured or Unstructured*

| | Unstructured Storage | Structured Storage |
|---|---|---|
| SQL Constraint Support | SQL constraints are not currently available. | SQL constraints are supported. |
| Indexing Support | Text and function-based indexes. | B-Tree, text and function-based indexes. |
| Optimized Memory Management | XML operations on the document require creating a DOM from the document. | XML operations can be optimized to reduce memory requirements. |

Much valuable information in an organization is in the form of semistructured and unstructured data. Typically this data is in files stored on a file server or in a CLOB column inside a database. The information in these files is in proprietary- or application-specific formats. It can only be accessed through specialist tools, such as word processors or spreadsheets, or programmatically using complex, proprietary APIs. Searching across this information is limited to facilities provided by a crawler or full-text indexing.

Major reasons for the rapid adoption of XML are that it allows for:

- Stronger data management
- More open access to semistructured and unstructured content.

Replacing proprietary file formats with XML allows organizations to achieve much higher levels of reuse of their semistructured and unstructured data. The content can be accurately described using XML schemas. The content can be easily accessed and updated using standard APIs based on DOM and XPath.

For example, information contained in an Excel spreadsheet is only accessible to the Excel program, or to a program that uses Microsoft COM APIs. The same information, stored in an XML document is accessible to any tool that can leverage the XML programming model. Structured data on the other hand does not suffer from these limitations. Structured data is typically stored as rows in tables within a relational database. These tables are accessed and searched using the relational model and the power and openness of SQL from a variety of tools and processing engines.

## XML/SQL Duality

A key objective of Oracle XML DB is to provide XML/ SQL duality. This means that the XML programmer can leverage the power of the relational model when working with XML content and the SQL programmer can leverage the flexibility of XML when working with relational content. This provides application developers with maximum flexibility, allowing them to use the most appropriate tools for a particular business problem.

**Relational and XML Metaphors are Interchangeable:** Oracle XML DB erases the traditional boundary between applications that work with structured data and those that work with semistructured and unstructured content. With Oracle XML DB the relational and XML metaphors become interchangeable.

XML/SQL duality means that the same data can be exposed as rows in a table and manipulated using SQL or exposed as nodes in an XML document and manipulated using techniques such as DOM or XSL transformation. Access and processing techniques are totally independent of the underlying storage format.

These features provide new, simple solutions to common business problems. For example:

- Relational data can quickly and easily be converted into HTML pages. Oracle XML DB provides new SQL functions that make it possible to generate XML directly from a SQL query. The XML can be transformed into other formats, such as HTML using the database-resident XSLT processor.
- You can easily leverage all of the information contained in their XML documents without the overhead of converting back and forth between different formats. With Oracle XML DB you can access XML content using SQL queries, On-line Analytical Processing (OLAP), and Business-Intelligence/Data Warehousing operations.
- Text, spatial data, and multimedia operations can be performed on XML Content.

## SQL/XML INCITS Standard SQL Functions

Oracle XML DB provides the SQL functions defined in the SQL/XML standard. The SQL/XML standard is defined by specifications prepared by the International Committee for Information Technology Standards (INCITS) Technical Committee H2. INCITS is the main standards body for developing standards for the syntax and semantics of database languages, including SQL.

The SQL/XML standard is an evolving standard, so the syntax and semantics of its functions are subject to change in the future. The Oracle XML DB implementation of SQL/XML functions will evolve accordingly.

SQL/XML functions fall into two categories:

- Functions that make it possible to *query and access XML* content as part of normal SQL operations.
- Functions that provide a standard way of *generating XML* from the result of a SQL `SELECT` statement.

With the SQL/XML functions you can address XML content in any part of a SQL statement. They use XPath notation to traverse the XML structure and identify the node or nodes on which to operate. The ability to embed XPath expressions in SQL statements greatly simplifies XML access. The following describes briefly some of the more important SQL/XML functions:

- **existsNode** – This is used in the `WHERE` clause of a SQL statement to restrict the set of documents returned by a query. The `existsNode` SQL function takes an XPath expression and applies it to an XML document. The function returns true (`1`) or false (`0`), depending on whether or not the document contains a node that matches the XPath expression.
- **extract** – This takes an XPath expression and returns the nodes that match the expression, as an XML document or fragment. If only a single node matches the XPath expression, then the result is a well-formed XML document. If multiple nodes match the XPath expression, then the result is a document fragment.
- **extractValue** – This takes an XPath expression and returns the corresponding leaf node. The XPath expression passed to `extractValue` should identify a single attribute or an element that has precisely one text node child. The result is returned in the appropriate SQL data type. Function `extractValue` is essentially a shortcut for `extract` plus either `getStringVal()` or `getNumberVal()`.
- **updateXML** – This allows partial updates to be made to an XML document, based on a set of XPath expressions. Each XPath expression identifies a target node in the document, and a new value for that node. SQL function `updateXML` allows multiple updates to be specified for a single XML document.

- **XMLSequence** – This makes it possible to expose the members of a collection as a virtual table

**See Also:**

- `http://www.incits.org/tc_home/h2.htm` for information on INCITS Technical Committee H2
- `http://www.w3.org/TR/xpath` for the XPath recommendation
- Chapter 4, "XMLType Operations" for detailed descriptions of the SQL/XML standard functions for *querying* XML data
- Generating XML Using SQL Functions on page 16-3 for detailed descriptions of the SQL/XML standard functions for *generating* XML data
- Chapter 3, "Using Oracle XML DB" for *examples* that use the SQL/XML standard functions

## Rewriting of XPath Expressions: XPath Rewrite

The SQL/XML SQL functions and their corresponding `XMLType` methods allow XPath expressions to be used to search collections of XML documents and to access a subset of the nodes contained within an XML document.

**See Also:** "Generating XML Using SQL Functions" on page 16-3 for information on SQL/XML functions

### How XPath Expressions are Evaluated by Oracle XML DB

Oracle XML DB provides two ways of evaluating XPath expressions that operate on `XMLType` columns and tables, depending on the XML storage method used:

- **Structured-storage XML data:** Oracle XML DB attempts to translate the XPath expression in a SQL/XML function into an equivalent SQL query. The SQL query references the object-relational data structures that underpin a schema-based `XMLType`. This process is referred to as **XPath rewrite.** It can occur when performing queries and `UPDATE` operations.
- **Unstructured-storage XML data:** Oracle XML DB evaluates the XPath expression using functional evaluation. Functional evaluation builds a DOM tree for each XML document, and then resolves the XPath programmatically using the methods provided by the DOM API. If the operation involves updating the DOM tree, the entire XML document has to be written back to disc when the operation is completed.

### Efficient Processing of SQL That Contains XPath Expressions

Oracle XML DB can rewrite SQL statements that contain XPath expressions to purely relational SQL statements, which can be processed efficiently. In this way, Oracle XML DB insulates the database optimizer from having to understand XPath notation and the XML data model. The database optimizer simply processes the rewritten SQL statement in the same manner as other SQL statements.

This means that the database optimizer can derive an execution plan based on conventional relational algebra. This allows Oracle XML DB to leverage all the features of the database and ensure that SQL statements containing XPath expressions are executed in a highly performant and efficient manner. To sum up, there is little

overhead with XPath rewrites, and Oracle XML DB can execute XPath-based queries at near-relational speed, while preserving the XML abstraction.

### When Can XPath Rewrite Occur?

XPath rewrite is possible when:

- A SQL statement contains SQL/XML SQL functions or `XMLType` methods that use XPath expressions to refer to one or more nodes within a set of XML documents.
- An `XMLType` column or table containing the XML documents is associated with a registered XML schema.
- An `XMLType` column or table uses structured storage techniques to provide the underlying storage model.
- The nodes referenced by an XPath expression can be mapped, using the XML schema, to attributes of the underlying SQL object model.

### What is the XPath-Rewrite Process?

XPath rewrite performs the following tasks:

1. Identify the set of XPath expressions included in the SQL statement.
2. Translate each XPath expression into an object relational SQL expression that references the tables, types, and attributes of the underlying SQL: 1999 object model.
3. Rewrite the original SQL statement into an equivalent object relational SQL statement.
4. Pass the new SQL statement to the database optimizer for plan generation and query execution.

In certain cases, XPath rewrite is not possible. This normally occurs when there is no SQL equivalent of the XPath expression. In this situation Oracle XML DB performs a functional evaluation of the XPath expressions.

In general, functional evaluation of a SQL statement is more expensive than XPath rewrite, particularly if the number of documents that needs to be processed is large. However the major advantage of functional evaluation is that it is always possible, regardless of whether or not the `XMLType` is stored using structured storage and regardless of the complexity of the XPath expression. When documents are stored using unstructured storage (in a `CLOB` value), functional evaluation is necessary any time SQL functions except `existsNode` are used. Function `existsNode` will also result in functional evaluation unless a `CTXXPATH` index or function-based index can be used to resolve the query.

Understanding the concept of XPath rewrite, and the conditions under which it can take place, is a key step in developing Oracle XML DB applications that will deliver the required levels of scalability and performance.

**See Also:** Chapter 6, "XPath Rewrite"

## Oracle XML DB Benefits

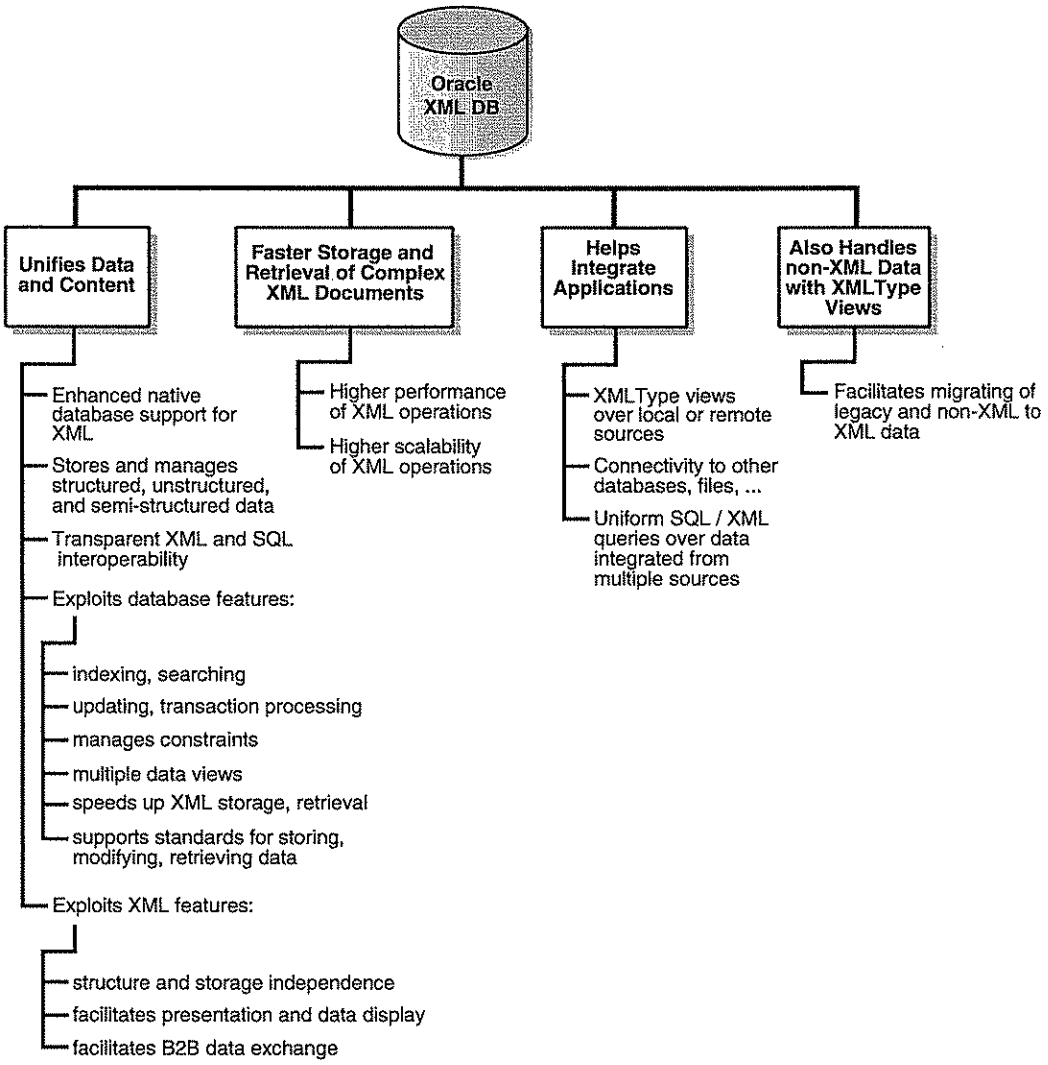
The following sections describe several benefits for using Oracle XML DB advantages including:

- Unifying Data and Content with Oracle XML DB
- Oracle XML DB Offers Faster Storage and Retrieval of Complex XML Documents

- Oracle XML DB Helps You Integrate Applications
- When Your Data Is Not XML You Can Use XMLType Views

Figure 1–6 summarizes the Oracle XML DB benefits.

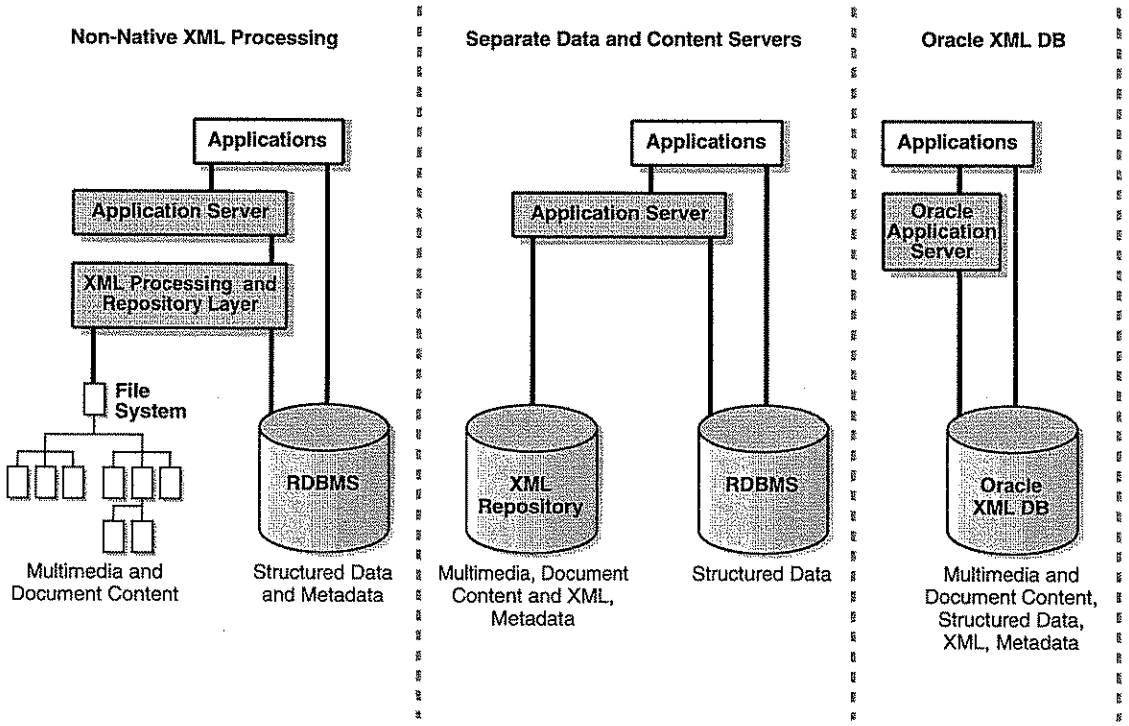
***Figure 1–6 Oracle XML DB Benefits***




## Unifying Data and Content with Oracle XML DB

Most application data and Web content is stored in a relational database or a file system, or both. XML is often used for transport, and it is generated from a database or a file system. As the volume of XML transported grows, the cost of regenerating these XML documents grows, and these storage methods become less effective at accommodating XML content.

*Figure 1–7 Unifying Data and Content: Some Common XML Architectures*




Organizations today typically manage their structured data and unstructured data differently:

- Unstructured data, in tables, makes document access transparent and table access complex
- Structured data, often in binary large objects (such as in BLOBs), makes access more complex and table access transparent.

With Oracle XML DB, you can store and manage data that is structured, unstructured, and semistructured using a standard data model and standard SQL and XML. You can perform SQL operations on XML documents and XML operations on object-relational (such as table) data.

### Exploiting Database Capabilities

Oracle Database has strong XML support with the following key capabilities:

- *Indexing and Search*: Applications use queries such as "find all the product definitions created between March and April 2002", a query that is typically supported by a B*Tree index on a date column. Oracle XML DB can enable efficient structured searches on XML data, saving content-management vendors the need to build proprietary query APIs to handle such queries. See Chapter 4, "XMLType Operations", Chapter 10, "Full-Text Search Over XML", and Chapter 16, "Generating XML Data from the Database".
- *Updates and Transaction Processing*: Commercial relational databases use fast updates of subparts of records, with minimal contention between users trying to update. As traditionally document-centric data participate in collaborative environments through XML, this requirement becomes more important. File or CLOB storage cannot provide the granular concurrency control that Oracle XML DB does. See Chapter 4, "XMLType Operations".

- *Managing Relationships*: Data with any structure typically has foreign key constraints. Currently, XML data-stores lack this feature, so you must implement any constraints in application code. Oracle XML DB enables you to constrain XML data according to XML schema definitions and hence achieve control over relationships that structured data has always enjoyed. See Chapter 5, "XML Schema Storage and Query: Basic" and the purchase-order examples in Chapter 4, "XMLType Operations".
- *Multiple Views of Data*: Most enterprise applications need to group data together in different ways for different modules. This is why relational views are necessary—to allow for these multiple ways to combine data. By allowing views on XML, Oracle XML DB creates different logical abstractions on XML for, say, consumption by different types of applications. See Chapter 18, "XMLType Views".
- *Performance and Scalability*: Users expect data storage, retrieval, and query to be fast. Loading a file or `CLOB` value, and parsing, are typically slower than relational data access. Oracle XML DB dramatically speeds up XML storage and retrieval. See Chapter 2, "Getting Started with Oracle XML DB" and Chapter 3, "Using Oracle XML DB".
- *Ease of Development*: Databases are foremost an application platform that provides standard, easy ways to manipulate, transform, and modify individual data elements. While typical XML parsers give standard read access to XML data they do not provide an easy way to modify and store individual XML elements. Oracle XML DB supports a number of standard ways to store, modify, and retrieve data: using XML Schema, XPath, DOM, and Java.

  **See Also:**

  - Chapter 13, "Java API for XMLType"
  - Chapter 22, "SQL Access Using RESOURCE_VIEW and PATH_VIEW"
  - Chapter 23, "PL/SQL Access Using DBMS_XDB"

### Exploiting XML Capabilities

If the drawbacks of XML file storage force you to break down XML into database tables and columns, there are several XML advantages you have left:

- *Structure Independence*: The open content model of XML cannot be captured easily in the pure tables-and-columns world. XML schemas allow global element declarations, not just scoped to a container. Hence you can find a particular data item regardless of where in the XML document it moves to as your application evolves. See Chapter 5, "XML Schema Storage and Query: Basic".
- *Storage Independence*: When you use relational design, your client programs must know where your data is stored, in what format, what table, and what the relationships are among those tables. `XMLType` enables you to write applications without that knowledge and allows DBAs to map structured data to physical table and column storage. See Chapter 5, "XML Schema Storage and Query: Basic" and Chapter 20, "Accessing Oracle XML DB Repository Data".
- *Ease of Presentation*: XML is understood natively by Web browsers, many popular desktop applications, and most Internet applications. Relational data is not generally accessible directly from applications; programming is required to make relational data accessible to standard clients. Oracle XML DB stores data as XML and makes it available as XML outside the database; no extra programming is required to display database content. See:

- Chapter 9, "Transforming and Validating XMLType Data".
- Chapter 16, "Generating XML Data from the Database".
- Chapter 18, "XMLType Views".
- *Oracle XML Developer's Kit Programmer's Guide*, in the chapter, "XSQL Pages Publishing Framework". It includes `XMLType` examples.

- ***Ease of Interchange***: XML is the language of choice in business-to-business (B2B) data exchange. If you are forced to store XML in an arbitrary table structure, you are using some kind of proprietary translation. Whenever you translate a language, information is lost and interchange suffers. By natively understanding XML and providing DOM fidelity in the storage/retrieval process, Oracle XML DB enables a clean interchange. See:
  - Chapter 9, "Transforming and Validating XMLType Data"
  - Chapter 18, "XMLType Views"

## Oracle XML DB Offers Faster Storage and Retrieval of Complex XML Documents

Users today face a performance barrier when storing and retrieving complex, large, or many XML documents. Oracle XML DB provides very high performance and scalability for XML operations. The major performance features are:

- Native `XMLType`. See Chapter 4, "XMLType Operations".
- The lazily evaluated virtual DOM support. See Chapter 11, "PL/SQL API for XMLType".
- Database-integrated XPath and XSLT support. This support is described in several chapters, including Chapter 4, "XMLType Operations" and Chapter 9, "Transforming and Validating XMLType Data".
- XML schema-caching support. See Chapter 5, "XML Schema Storage and Query: Basic".
- CTXPATH Text indexing. See Chapter 10, "Full-Text Search Over XML".
- The hierarchical index over Oracle XML DB Repository. See Chapter 20, "Accessing Oracle XML DB Repository Data".

## Oracle XML DB Helps You Integrate Applications

Oracle XML DB enables data from disparate systems to be accessed through gateways and combined into one common data model. This reduces the complexity of developing applications that must deal with data from different stores.

## When Your Data Is Not XML You Can Use XMLType Views

`XMLType` views provide a way for you wrap existing relational and object-relational data in XML format. This is especially useful if, for example, your legacy data is not in XML but you need to migrate to an XML format. Using `XMLType` views you do not need to alter your application code.

**See Also:** Chapter 18, "XMLType Views"

To use `XMLType` views, you must first register an XML schema with annotations that represent the bi-directional mapping from XML to SQL object types and back to XML. An `XMLType` view conforming to this schema (mapping) can then be created by

providing an underlying query that constructs instances of the appropriate SQL object type. Figure 1–6 summarizes the Oracle XML DB advantages.

## Searching XML Data Stored in CLOBs Using Oracle Text

Oracle enables special indexing on XML, including Oracle Text indexes for section searching, special SQL functions to process XML, aggregation of XML, and special optimization of queries involving XML.

XML data stored in Character Large Objects (`CLOB` datatype) or stored in `XMLType` columns in structured storage (object-relationally), can be indexed using Oracle Text. SQL functions `hasPath` and `inPath` are designed to optimize XML data searches where you can search within XML text for substring matches.

Oracle XML DB also provides:

- SQL function `contains` and XPath function `ora:contains`, which can be used with SQL function `existsNode` for XPath-based searches.
- The ability to create indexes on `URIType` and `XDBURIType` columns.
- Index type `CTXXPATH`, which allows higher performance XPath searching using `existsNode`.

**See Also:**

- Chapter 10, "Full-Text Search Over XML"
- *Oracle Text Application Developer's Guide*
- *Oracle Text Reference*

## Building Messaging Applications using Oracle Streams Advanced Queuing

Oracle Streams Advanced Queuing supports the use of:

- `XMLType` as a message/payload type, including XML schema-based `XMLType`
- Queuing or dequeuing of `XMLType` messages

**See Also:**

- *Oracle Streams Advanced Queuing User's Guide and Reference* for information about using `XMLType` with Oracle Streams Advanced Queuing
- Chapter 31, "Exchanging XML Data with Oracle Streams AQ"

## Requirements for Running Oracle XML DB

Oracle XML DB is available with Oracle9*i* release 2 (9.2) and higher.

**See Also:**

- `http://www.oracle.com/technology/tech/xml/` for the latest news and white papers on Oracle XML DB
- Chapter 2, "Getting Started with Oracle XML DB"

## Standards Supported by Oracle XML DB

Oracle XML DB supports all major XML, SQL, Java, and Internet standards:

- W3C XML Schema 1.0 Recommendation. You can register XML schemas, validate stored XML content against XML schemas, or constrain XML stored in the server to XML schemas.
- W3C XPath 1.0 Recommendation. You can search or traverse XML stored inside the database using XPath, either from HTTP(S) requests or from SQL.
- ISO-ANSI Working Draft for XML-Related Specifications (SQL/XML) [ISO/IEC 9075 Part 14 and ANSI]. You can use the emerging ANSI SQL/XML functions to query XML from SQL. The task force defining these specifications falls under the auspices of the International Committee for Information Technology Standards (INCITS). The SQL/XML specification will be fully aligned with SQL:2003. SQL/XML functions are sometimes referred to as SQLX functions.
- Java Database Connectivity (JDBC) API. JDBC access to XML is available for Java programmers.
- W3C XSL 1.0 Recommendation. You can transform XML documents at the server using XSLT.
- W3C DOM Recommendation Levels 1.0 and 2.0 Core. You can retrieve XML stored in the server as an XML DOM, for dynamic access.
- Protocol support. You can store or retrieve XML data from Oracle XML DB using standard protocols such as HTTP(S), FTP, and IETF WebDAV, as well as Oracle Net.
- Java Servlet version 2.2, (except that the servlet WAR file, `web.xml` is not supported in its entirety, and only one `ServletContext` and one `web-app` are currently supported, and stateful servlets are not supported).
- Simple Object Access Protocol (SOAP). You can access XML stored in the server from SOAP requests. You can build, publish, or find Web Services using Oracle XML DB and Oracle9*i*AS, using WSDL and UDDI. You can use Oracle Streams Advanced Queuing IDAP, the SOAP specification for queuing operations, on XML stored in Oracle Database.

**See Also:**

- "SQL/XML INCITS Standard SQL Functions" for more information on the ANSI SQL/XML functions
- Chapter 25, "FTP, HTTP(S), and WebDAV Access to Repository Data" for more information on protocol support
- Chapter 27, "Writing Oracle XML DB Applications in Java" for information on using the Java servlet
- Chapter 31, "Exchanging XML Data with Oracle Streams AQ" and *Oracle Streams Advanced Queuing User's Guide and Reference.* for information on using SOAP

## Oracle XML DB Technical Support

Besides your regular channels of support through your customer representative or consultant, technical support for Oracle Database XML-enabled technologies is available free through the Discussions option on Oracle Technology Network (OTN):

```
http://www.oracle.com/technology/tech/xml/
```

## Oracle XML DB Examples Used in This Manual

This manual contains examples that illustrate the use of Oracle XML DB and `XMLType`. The examples are based on a number of database schema, sample XML documents, and sample XML schema.

> **See Also:** Appendix D, "Oracle-Supplied XML Schemas and Examples"

## Further Oracle XML DB Case Studies and Demonstrations

Visit OTN to view Oracle XML DB examples, white papers, case studies, and demonstrations.

### Oracle XML DB Examples and Tutorials

You can peruse more Oracle XML DB examples on OTN:

```
http://www.oracle.com/technology/tech/xml/
```

Comprehensive XML classes on how to use Oracle XML DB are also available. See the Oracle University link on OTN.

### Oracle XML DB Case Studies and Demonstrations

Several detailed Oracle XML DB case studies are available on OTN and include the following:

- Oracle XML DB Downloadable Demonstration. This detailed demonstration illustrates how to use many Oracle XML DB features. Parts of this demonstration are also included in Chapter 3, "Using Oracle XML DB".
- Content Management System (CMS) application. This illustrates how you can store files on the database using Oracle XML DB Repository in hierarchically organized folders, place the files under version control, provide security using ACLs, transform XML content to a desired format, search content using Oracle Text, and exchange XML messages using Oracle Streams Advanced Queueing (to request privileges on files or for sending externalization requests). See `http://www.oracle.com/technology/sample_code/tech/xml/xmldb/cmsxdb/content.html`.
- XML Dynamic News. This is a complete J2EE 1.3 based application that demonstrates Java and Oracle XML DB features for an online news portal. News feeds are stored and managed persistently in Oracle XML DB. Various other news portals can customize this application to provide static or dynamic news services to end users. End users can personalize their news pages by setting their preferences. The application also demonstrates the use of Model View Controller (MVC) architecture and various J2EE design patterns. See `http://www.oracle.com/technology/sample_code/tech/xml/xmlnews/content.html`
- SAX Loader Application. This demonstrates an efficient way to break up large files containing multiple XML documents outside the database and insert them into the database as a set of separate documents. This is provided as a standalone and a web-based application.
- Oracle XML DB Utilities package. This highlights the subprograms provided with the `XDB_Utilities` package. These subprograms operate on `BFILE` values,

CLOB values, DOM, and Oracle XML DB Resource APIs. With this package, you can perform basic Oracle XML DB foldering operations, read and load XML files into a database, and perform basic DOM operations through PL/SQL.

- Card Payment Gateway Application. This application uses Oracle XML DB to store all your data in XML format and enables access to the resulting XML data using SQL. It illustrates how a credit card company can store its account and transaction data in the database and also maintain XML fidelity.
- Survey Application. This application determines what members want from Oracle products. OTN posts the online surveys and studies the responses. This Oracle XML DB application demonstrates how a company can create dynamic, interactive HTML forms, deploy them to the Internet, store the responses as XML, and analyze them using the XML enabled Oracle Database.

# 3

# Using Oracle XML DB

This chapter provides an overview of how to use Oracle XML DB. The examples here illustrate techniques for accessing and managing XML content in purchase orders. The format and data of XML purchase orders are well suited for Oracle XML DB storage and processing techniques because purchase orders are highly structured XML documents. However, the majority of techniques introduced here can also be used to manage other types of XML documents, such as containing unstructured or semistructured data. This chapter also further explains Oracle XML DB concepts introduced in Chapter 1, "Introduction to Oracle XML DB".

This chapter contains these topics:

- Storing XML as XMLType
- Creating XMLType Tables and Columns
- Loading XML Content into Oracle XML DB
- Overview of the W3C XML Schema Recommendation
- Using XML Schema with Oracle XML DB
- Identifying XML Schema Instance Documents
- Using the Database to Enforce XML Data Integrity
- DML Operations on XML Content Using Oracle XML DB
- Querying XML Content Stored in Oracle XML DB
- Relational Access to XML Content Stored in Oracle XML DB Using Views
- Updating XML Content Stored in Oracle XML DB
- Namespace Support in Oracle XML DB
- Processing XMLType Methods and XML-Specific SQL Functions
- Understanding and Optimizing XPath Rewrite
- Accessing Relational Database Content Using XML
- XSL Transformation and Oracle XML DB
- Using Oracle XML DB Repository
- Viewing Relational Data as XML From a Browser
- XSL Transformation Using DBUri Servlet

# Storing XML as XMLType

Before the introduction of Oracle XML DB, there were two ways to store XML content in Oracle Database:

- Use Oracle XML Developer's Kit (XDK) to parse the XML document outside Oracle Database, and store the extracted XML data as rows in one or more tables in the database.
- Store the XML document in Oracle Database using a Character Large Object (CLOB), Binary Large Object (BLOB), Binary File (BFILE), or VARCHAR column.

In both cases, Oracle Database is unaware that it is managing XML content.

The introduction of Oracle XML DB and the XMLType datatype provides new techniques that facilitate the persistence of XML content in the database. These techniques include the ability to store XML documents in an XMLType column or table, or in Oracle XML DB Repository. Storing XML as an XMLType column or table makes Oracle Database aware that the content is XML. This allows the database to:

- Perform XML-specific validations, operations, and optimizations on the XML content
- Facilitate highly efficient processing of XML content by Oracle XML DB

## What is XMLType?

Oracle9*i* release 1 (9.0.1) introduced a new datatype, XMLType, to facilitate native handling of XML data in the database:

- XMLType can represent an XML document in the database, so it is accessible in SQL.
- XMLType has built-in methods that operate on XML content. For example, you can use XMLType methods to create, extract, and index XML data stored in Oracle Database.
- XMLType functionality is also available through a set of Application Program Interfaces (APIs) provided in PL/SQL and Java.
- XMLType can be used in PL/SQL stored procedures for parameters, return values, and variables

With XMLType, SQL developers can leverage the power of the relational database while working in the context of XML. XML developers can leverage the power of XML standards while working in the context of a relational database.

XMLType can be used as the datatype of columns in tables and views. XMLType variables can be used in PL/SQL stored procedures as parameters and return values. You can also use XMLType in SQL, PL/SQL, C, Java (through JDBC), and Oracle Data Provider for .NET (ODP.NET).

The XMLType API provides a number of useful methods that operate on XML content. For example, method `extract()` extracts one or more nodes from an XMLType instance. Many of these XMLType methods are also provided as SQL functions. For example, SQL function `extract` corresponds to XMLType method `extract()`.

Oracle XML DB functionality is based on the Oracle XML Developer's Kit C implementations of the relevant XML standards such as XML Parser, XML DOM, and XML Schema Validator.

## Benefits of the XMLType Datatype and API

The XMLType datatype and application programming interface (API) enable SQL operations on XML content and XML operations on SQL content:

- *Versatile API.* XMLType has a versatile API for application development that includes built-in functions, indexing, and navigation support.
- *XMLType and SQL.* You can use XMLType in SQL statements, combined with other datatypes. For example, you can query XMLType columns and join the result of the extraction with a relational column. Oracle Database determines an optimal way to run such queries.
- *Indexing:*
  - Oracle XML DB lets you create *B*Tree indexes* on the object-relational tables that provide structured storage of XMLType tables and columns.
  - Oracle Text indexing supports *text indexing* of the content of structured and unstructured XMLType tables and columns.
  - The CTXXPATH domain index type of Oracle Text provides an *XML-specific text index* with transactional semantics. This index type can speed up certain XPath-based searches on both structured and unstructured content.
  - *Function-based indexes* can be used to create indexes on explicit XPath expressions for both structured and unstructured XMLType storage.

## When to Use XMLType

Use XMLType any time you want to use the database as a persistent storage of XML. XMLType functionality includes the following:

- *SQL queries on part of or the whole XML document* – SQL functions existsNode and extract provide the necessary SQL query functions over XML documents.
- *XPath access using SQL functions* existsNode *and* extract – XMLType uses the built-in C XML parser and processor and hence provides better performance and scalability when used inside the server.
- *Strong typing inside SQL statements and PL/SQL functions* – The strong typing offered by XMLType ensures that the values passed in are XML values and not any arbitrary text string.
- *Indexing on XPath document queries* – XMLType has methods that you can use to create function-based indexes that optimize searches.
- *Separation of applications from storage models* – Using XMLType instead of CLOB values or relational storage allows applications to gracefully move to various storage alternatives later without affecting any of the query or DML statements in the application.
- *Support for future optimizations* – New XML functionality will support XMLType. Because Oracle Database is natively aware that XMLType can store XML data, better optimizations and indexing techniques can be done. By writing applications to use XMLType, these optimizations and enhancements can be easily achieved and preserved in future releases without your needing to rewrite applications.

## Two Ways to Store XMLType Data: LOBs and Structured

XMLType data can be stored in two ways:

- *In Large Objects (LOBs)* – LOB storage maintains content fidelity, also called document fidelity. The original XML is preserved, including whitespace. An entire XML document is stored as a whole in a LOB. For non-schema-based storage, `XMLType` offers a Character Large Object (CLOB) storage option.
- *In Structured storage (in tables and views)* – Structured storage maintains DOM (Document Object Model) fidelity.

Native `XMLType` instances contain hidden columns that store this extra information that does not fit into the SQL object model. This information can be accessed through APIs in SQL or Java, using functions such as `extractNode()`.

Changing `XMLType` storage from structured storage to LOB, or vice versa, is possible using database `IMPORT` and `EXPORT`. Your application code does not need to change. You can then change XML storage options when tuning your application, because each storage option has its own benefits.

## Advantages and Disadvantages of XML Storage Options in Oracle XML DB

Table 3–1 summarizes some advantages and disadvantages to consider when selecting your Oracle XML DB storage option. Storage options are also discussed in Table 1–3, " XML Storage Options: Structured or Unstructured" and Chapter 2, "Getting Started with Oracle XML DB".

***Table 3–1 XML Storage Options in Oracle XML DB***

| Feature | LOB Storage (with Oracle Text Index) | Structured Storage (with B*Tree index) |
|---|---|---|
| Database schema flexibility | Very flexible when schemas change. | Limited flexibility for schema changes. Similar to the `ALTER TABLE` restrictions. |
| Data integrity and accuracy | Maintains the original XML content fidelity, important in some applications. | Trailing new lines, whitespace within tags, and data format is lost. DOM fidelity is maintained. |
| Performance | Mediocre performance for DML. | Excellent DML performance. |
| Access to SQL | Some accessibility to SQL features. | Good accessibility to existing SQL features, such as constraints, indexes, and so on |
| Space needed | Can consume considerable space. | Needs less space in particular when used with an Oracle XML DB registered XML schema. |

## When to Use CLOB Storage for XMLType

Use `CLOB` storage for `XMLType` in the following cases:

- When you are interested in storing and retrieving the whole document.
- When you do not need to perform piece-wise updates on XML documents.

# Creating XMLType Tables and Columns

The following examples create `XMLType` columns and tables for managing XML content in Oracle Database:

***Example 3–1 Creating a Table with an XMLType Column***

```
CREATE TABLE mytable1 (key_column VARCHAR2(10) PRIMARY KEY,
                       xml_column XMLType);

Table created.
```

***Example 3–2 Creating a Table of XMLType***

```
CREATE TABLE mytable2 OF XMLType;

Table created.
```

# Loading XML Content into Oracle XML DB

You can load XML content into Oracle XML DB using several techniques:

- Table-based loading:
  - Loading XML Content Using SQL or PL/SQL
  - Loading XML Content Using Java
  - Loading XML Content Using C
  - Loading Large XML Files That Contain Small XML Documents
  - Loading Large XML Files Using SQL*Loader
- Path-based repository loading techniques:
  - Loading XML Documents into the Repository Using DBMS_XDB
  - Loading Documents into the Repository Using Protocols

## Loading XML Content Using SQL or PL/SQL

You can use a simple `INSERT` operation in SQL or PL/SQL to load an XML document into the database. Before the document can be stored as an `XMLType` column or table, it must be converted into an `XMLType` instance using one of the `XMLType` constructors.

**See Also:**

- Chapter 4, "XMLType Operations"
- "APIs for XML" on page 1-5
- *Oracle Database PL/SQL Packages and Types Reference* for a description of the `XMLType` constructors

`XMLType` **constructors** allow an `XMLType` instance to be created from different sources, including `VARCHAR`, `CLOB`, and `BFILE` values. The constructors accept additional arguments that reduce the amount of processing associated with `XMLType` creation. For example, if you are sure that a given source XML document is valid, you can provide an argument to the constructor that disables the type-checking that is otherwise performed.

In addition, if the source data is not encoded in the database character set, an `XMLType` instance can be constructed using a `BFILE` or `BLOB` value. The encoding of the source data is specified through the character set id (`csid`) argument of the constructor.

### Create a SQL Directory That Points to the Needed Directory

Example 3–3 shows how to insert XML content into an `XMLType` table. Before making this insertion, you must create a SQL directory object that points to the directory containing the file to be processed. To do this, you must have the `CREATE ANY DIRECTORY` privilege.

**See Also:** *Oracle Database SQL Reference*, Chapter 18, under GRANT

```
CREATE DIRECTORY xmldir AS path_to_folder_containing_XML_file';
```

***Example 3–3 Inserting XML Content into an XMLType Table***

```
INSERT INTO mytable2 VALUES (XMLType(bfilename('XMLDIR', 'purchaseOrder.xml'),
                                     nls_charset_id('AL32UTF8')));

1 row created.
```

The value passed to `nls_charset_id()` indicates that the encoding for the file to be read is UTF-8.

## Loading XML Content Using Java

***Example 3–4 Inserting XML Content into an XML Type Table Using Java***

This example shows how to load XML content into Oracle XML DB by first creating an `XMLType` instance in Java, given a Document Object Model (DOM).

```
public void doInsert(Connection conn, Document doc)
throws Exception
{
   String SQLTEXT = "INSERT INTO purchaseorder VALUES (?)";
   XMLType xml = null;
   xml = XMLType.createXML(conn,doc);
   OraclePreparedStatement sqlStatement = null;
   sqlStatement = (OraclePreparedStatement) conn.prepareStatement(SQLTEXT);
   sqlStatement.setObject(1,xml);
   sqlStatement.execute();
}

1 row selected.
```

The "Simple Bulk Loader Application" available on the Oracle Technology Network (OTN) site at `http://www.oracle.com/technology/sample_code/tech/xml/xmldb/content.html` demonstrates how to load a directory of XML files into Oracle XML DB using Java Database Connectivity (JDBC). JDBC is a set of Java interfaces to Oracle Database.

## Loading XML Content Using C

Example 3–5 shows, in C, how to insert XML content into an `XMLType` table by creating an `XMLType` instance given a DOM.

***Example 3–5 Inserting XML Content into an XMLType Table Using C***

```
#include "stdio.h"
#include <xml.h>
#include <stdlib.h>
#include <string.h>
#include <ocixmldb.h>
OCIEnv *envhp;
OCIError *errhp;
OCISvcCtx *svchp;
OCIStmt *stmthp;
```

```
OCIServer *srvhp;
OCIDuration dur;
OCISession *sesshp;
oratext *username = "QUINE";
oratext *password = "CURRY";
oratext *filename = "AMCEWEN-20021009123336171PDT.xml";
oratext *schemaloc = "http://localhost:8080/source/schemas/poSource/xsd/purchaseOrder.xsd";

/*---------------------------------------------------------*/
/* Execute a SQL statement that binds XML data             */
/*---------------------------------------------------------*/

sword exec_bind_xml(OCISvcCtx *svchp, OCIError *errhp, OCIStmt *stmthp,
                    void *xml,        OCIType *xmltdo, OraText *sqlstmt)
{
  OCIBind *bndhp1 = (OCIBind *) 0;
  sword  status = 0;
  OCIInd ind = OCI_IND_NOTNULL;
  OCIInd *indp = &ind;
  if(status = OCIStmtPrepare(stmthp, errhp, (OraText *)sqlstmt,
                             (ub4)strlen((const char *)sqlstmt),
                             (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT))
    return OCI_ERROR;
  if(status = OCIBindByPos(stmthp, &bndhp1, errhp, (ub4) 1, (dvoid *) 0,
                           (sb4) 0, SQLT_NTY, (dvoid *) 0, (ub2 *)0,
                           (ub2 *)0, (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT))
    return OCI_ERROR;
  if(status = OCIBindObject(bndhp1, errhp, (CONST OCIType *) xmltdo,
                            (dvoid **) &xml, (ub4 *) 0,
                            (dvoid **) &indp, (ub4 *) 0))
    return OCI_ERROR;
  if(status = OCIStmtExecute(svchp, stmthp, errhp, (ub4) 1, (ub4) 0,
                             (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                             (ub4) OCI_DEFAULT))
    return OCI_ERROR;
  return OCI_SUCCESS;
}

/*---------------------------------------------------------*/
/* Initialize OCI handles, and connect                     */
/*---------------------------------------------------------*/

sword init_oci_connect()
{
. . .
}

/*---------------------------------------------------------*/
/* Free OCI handles, and disconnect                        */
/*---------------------------------------------------------*/

void free_oci()
{
. . .
}

void main()
{
  OCIType *xmltdo;
  xmldocnode  *doc;
  ocixmldbparam params[1];
  xmlerr       err;
  xmlctx  *xctx;
  oratext *ins_stmt;
  sword    status;
  xmlnode *root;
```

```
oratext buf[10000];

/* Initialize envhp, svchp, errhp, dur, stmthp */
init_oci_connect();

/* Get an XML context */
params[0].name_ocixmldbparam = XCTXINIT_OCIDUR;
params[0].value_ocixmldbparam = &dur;
xctx = OCIXmlDbInitXmlCtx(envhp, svchp, errhp, params, 1);
if (!(doc = XmlLoadDom(xctx, &err, "file", filename,
                       "schema_location", schemaloc, NULL)))
  {
    printf("Parse failed.\n");
    return;
  }
else
  printf("Parse succeeded.\n");
root = XmlDomGetDocElem(xctx, doc);
printf("The xml document is :\n");
XmlSaveDom(xctx, &err, (xmlnode *)doc, "buffer", buf, "buffer_length", 10000, NULL);
printf("%s\n", buf);

/* Insert the document into my_table */
ins_stmt = (oratext *)"insert into purchaseorder values (:1)";
status = OCITypeByName(envhp, errhp, svchp, (const text *) "SYS",
                       (ub4) strlen((const char *)"SYS"), (const text *) "XMLTYPE",
                       (ub4) strlen((const char *)"XMLTYPE"), (CONST text *) 0,
                       (ub4) 0, OCI_DURATION_SESSION, OCI_TYPEGET_HEADER,
                       (OCIType **) &xmltdo);
if (status == OCI_SUCCESS)
  {
    status = exec_bind_xml(svchp, errhp, stmthp, (void *)doc,
                           xmltdo, ins_stmt);
  }
if (status == OCI_SUCCESS)
  printf ("Insert successful\n");
else
  printf ("Insert failed\n");

/* Free XML instances */
if (doc)
  XmlFreeDocument((xmlctx *)xctx, (xmldocnode *)doc);
/* Free XML CTX */
OCIXmlDbFreeXmlCtx(xctx);
free_oci();
}
```

**See Also:** Appendix D, "Oracle-Supplied XML Schemas and Examples" for a complete listing of this example

## Loading Large XML Files That Contain Small XML Documents

When loading large XML files consisting of a collection of smaller XML documents, it is often more efficient to use Simple API for XML (SAX) parsing to break the file into a set of smaller documents, and then insert those documents. SAX is an XML standard interface provided by XML parsers for event-based applications.

You can use SAX to load a database table from very large XML files in the order of 30 Mb or larger, by creating individual documents from a collection of nodes. You can also bulk load XML files.

**See Also:** `http://www.oracle.com/technology/sample_code/tech/xml/xmldb/content.html`, "SAX Loader Application" for an example of how to do this

## Loading Large XML Files Using SQL*Loader

Use SQL*Loader to load large amounts of XML data into Oracle Database. SQL*Loader loads in one of two modes, conventional or direct path. Table 3–2 compares these modes.

***Table 3–2 Comparing SQL*Loader Conventional and Direct Load Modes***

| Conventional Load Mode | Direct Path Load Mode |
|---|---|
| Uses SQL to load data into Oracle Database. This is the *default* mode. | Bypasses SQL and streams the data directly into Oracle Database. |
| *Advantage:* Follows SQL semantics. For example triggers are fired and constraints are checked. | *Advantage:* This loads data much faster than the conventional load mode. |
| *Disadvantage:* This loads data slower than with the direct load mode. | *Disadvantage:* SQL semantics are not obeyed. For example triggers are not fired and constraints are not checked. |

**See Also:**

- Chapter 29, "Loading XML Data Using SQL*Loader"
- Example 29–1, "Loading Very Large XML Documents Into Oracle Database Using SQL*Loader" on page 29-4 for an example of direct loading of XML data.

## Loading XML Documents into the Repository Using DBMS_XDB

You can also store XML documents in Oracle XML DB Repository, and access these documents using path-based rather than table-based techniques. To load an XML document into the repository under a given path, use PL/SQL package `DBMS_XDB`. This is illustrated by the following example.

***Example 3–6 Inserting XML Content into the Repository Using PL/SQL DBMS_XDB***

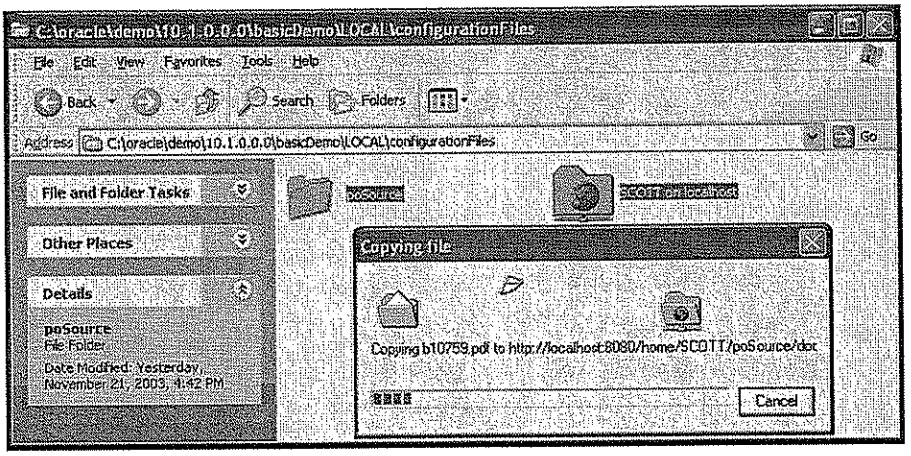
```
DECLARE
  res BOOLEAN;
BEGIN
  res := DBMS_XDB.createResource('/home/QUINE/purchaseOrder.xml',
                                 bfilename('XMLDIR', 'purchaseOrder.xml'),
                                 nls_charset_id('AL32UTF8'));
END;
/
```

Many operations for configuring and using Oracle XML DB are based on processing one or more XML documents – for example, registering an XML schema and performing an XSL transformation. The easiest way to make these XML documents available to Oracle Database is to load them into Oracle XML DB Repository.

## Loading Documents into the Repository Using Protocols

You can load XML documents from a local file system into Oracle XML DB Repository using protocols such as WebDAV, from Windows Explorer or other tools that support WebDAV. Figure 3–1 shows a simple drag and drop operation for copying the contents of the `SCOTT` folder from the local hard drive to the `poSource` folder in the Oracle XML DB Repository.

*Figure 3–1 Using Windows Explorer to Load Content into the Repository*




The copied folder might contain, for example, an XML schema document, an HTML page, and some XSLT style sheets.

---

**Note:** Oracle XML DB Repository can also store content that is not XML data, such as HTML files, JPEG images, word documents, as well as XML documents (schema-based and non-schema-based).

---

## Character Sets of XML Documents

This section describes how character sets of XML documents are determined.

---

**Caution:** *AL32UTF8* is the Oracle Database character set that is appropriate for `XMLType` data. It is equivalent to the IANA registered standard UTF-8 encoding, which supports all valid XML characters.

Do not confuse Oracle Database database character set UTF8 (no hyphen) with database character set AL32UTF8 or with character *encoding* UTF-8. Database character set UTF8 has been *superseded* by AL32UTF8. Do *not* use UTF8 for XML data. UTF8 supports only Unicode version 3.1 and earlier; it does not support all valid XML characters. AL32UTF8 has no such limitation.

Using database character set UTF8 for XML data could potentially *stop a system or affect security negatively*. If a character that is not supported by the database character set appears in an input-document element name, a replacement character (usually "?") will be substituted for it. This will terminate parsing and raise an exception. It could cause a fatal error.

---

### XML Encoding Declaration

Each XML document is composed of units called entities. Each entity in an XML document may use a different encoding for its characters. Entities that are stored in an encoding other than UTF-8 or UTF-16 must begin with a declaration containing an encoding specification indicating the character encoding in use. For example:

```
<?xml version='1.0' encoding='EUC-JP' ?>
```

Entities encoded in UTF-16 must begin with the Byte Order Mark (BOM), as described in Appendix F of the XML 1.0 Reference. For example, on big-endian platforms, the BOM required of a UTF-16 data stream is `#xFEFF`.

In the absence of both the encoding declaration and the BOM, the XML entity is assumed to be encoded in UTF-8. Because ASCII is a subset of UTF-8, ASCII entities do not require an encoding declaration.

In many cases, external sources of information are available, besides the XML data, to provide the character encoding in use. For example, the encoding of the data can be obtained from the `charset` parameter of the Content-Type field in an HTTP(S) request as follows:

```
Content-Type: text/xml; charset=ISO-8859-4
```

## Character-Set Determination When Loading XML Documents into the Database

In releases prior to Oracle Database 10*g* Release 1, all XML documents were assumed to be in the *database* character set, regardless of the document encoding declaration. With Oracle Database 10*g* Release 1, the document encoding is detected from the encoding declaration when the document is loaded into the database.

However, if the XML data is obtained from a `CLOB` or `VARCHAR` value, then the encoding declaration is *ignored*, because these two data types are always encoded in the database character set.

In addition, when loading data into Oracle XML DB, either through programmatic APIs or transfer protocols, you can provide external encoding to override the document encoding declaration. An error is raised if you try to load a schema-based XML document that contains characters that are not legal in the determined encoding.

The following examples show different ways to specify external encoding:

- Using PL/SQL function `DBMS_XDB.createResource` to create a file resource from a `BFILE`, you can specify the file encoding with the *`CSID`* argument. If a zero *`CSID`* is specified then the file encoding is auto-detected from the document encoding declaration.

```
CREATE DIRECTORY xmldir AS '/private/xmldir';
CREATE OR REPLACE PROCEDURE loadXML(filename VARCHAR2, file_csid NUMBER) IS
   xbfile  BFILE;
   RET     BOOLEAN;
BEGIN
   xbfile := bfilename('XMLDIR', filename);
   ret := DBMS_XDB.createResource('/public/mypurchaseorder.xml',
                                  xbfile,
                                  file_csid);
END;
/
```

- Use the FTP protocol to load documents into Oracle XML DB. Use the `quote set_charset` FTP command to indicate the encoding of the files to be loaded.

```
FTP> quote set_charset Shift_JIS
FTP> put mypurchaseorder.xml
```

- Use the HTTP(S) protocol to load documents into Oracle XML DB. Specify the encoding of the data to be transmitted to Oracle XML DB in the request header.

```
Content-Type: text/xml; charset= EUC-JP
```

## Character-Set Determination When Retrieving XML Documents from the Database

XML documents stored in Oracle XML DB can be retrieved using a SQL client, programmatic APIs, or transfer protocols. You can specify the encoding of the retrieved data (except in Oracle Database releases prior to 10*g*, where XML data is retrieved only in the database character set).

The character set for an XML document retrieved from the database is determined in the following ways:

- **SQL Client** – If a SQL client (such as SQL*Plus) is used to retrieve XML data, then the character set is determined by the client-side environment variable `NLS_LANG`. In particular, this setting overrides any explicit character-set declarations in the XML data itself.

  For example, if you set the client side `NLS_LANG` variable to `AMERICAN_AMERICA.AL32UTF8` and then retrieve an XML document with encoding `EUC_JP` provided by declaration `<?xml version="1.0" encoding="EUC-JP"?>`, the character set of the retrieved document is `AL32UTF8`, *not* `EUC_JP`.

  **See Also:** *Oracle Database Globalization Support Guide* for information on `NLS_LANG`

- **PL/SQL and APIs** – Using PL/SQL or programmatic APIs, you can retrieve XML data into `VARCHAR`, `CLOB`, or `XMLType` datatypes. As for SQL clients, you can control the encoding of the retrieved data by setting `NLS_LANG`.

  You can also retrieve XML data into a `BLOB` value using `XMLType` and `URIType` methods. These methods let you specify the character set of the returned `BLOB` value. Here is an example:

```
CREATE OR REPLACE FUNCTION getXML(pathname VARCHAR2, charset VARCHAR2)
                    RETURN BLOB IS
    xblob  BLOB;
BEGIN
    SELECT e.RES.getBlobVal(nls_charset_id(charset)) INTO xblob
      FROM RESOURCE_VIEW e WHERE ANY_PATH = pathname;
   RETURN xblob;
END;
/
```

- **FTP** – You can use the FTP `quote set_nls_locale` command to set the character set:

```
FTP> quote set_nls_locale EUC-JP
FTP> get mypurchaseorder.xml
```

  **See Also:** FTP Quote Methods on page 25-9

- **HTTP(S)** – You can use the `Accept-Charset` parameter in an HTTP(S) request:

```
/httptest/mypurchaseorder.xml  1.1 HTTP/Host: localhost:2345
Accept: text/*
Accept-Charset:  iso-8859-1, utf-8
```

  **See Also:** Controlling Character Sets for HTTP(S) on page 25-15

# Overview of the W3C XML Schema Recommendation

The W3C XML Schema Recommendation defines a standardized language for specifying the structure, content, and certain semantics of a set of XML documents. An XML schema can be considered the metadata that describes a class of XML documents. The XML Schema Recommendation is described at:
`http://www.w3.org/TR/xmlschema-0/`

## XML Instance Documents

Documents conforming to a given XML schema can be considered as members or instances of the class defined by that XML schema. Consequently the term *instance document* is often used to describe an XML document that conforms to a given XML schema. The most common use of an XML schema is to validate that a given instance document conforms to the rules defined by the XML schema.

## XML Schema for Schemas

The W3C Schema working group publishes an XML schema, often referred to as the "Schema for Schemas". This XML schema provides the definition, or vocabulary, of the XML Schema language. All valid XML schemas can be considered as members of the class defined by this XML schema. This means that an XML schema is an XML document that conforms to the class defined by the XML schema published at `http://www.w3.org/2001/XMLSchema`.

## Editing XML Schemas

XML schemas can be authored and edited using any of the following:

- A simple text editor, such as emacs or vi
- An XML schema-aware editor, such as the XML editor included with Oracle JDeveloper
- An explicit XML schema-authoring tool, such as XMLSpy from Altova Corporation

## XML Schema Features

The XML Schema language defines 47 scalar datatypes. This provides for strong typing of elements and attributes. The W3C XML Schema Recommendation also supports object-oriented techniques such as inheritance and extension, hence you can design XML schema with complex objects from base data types defined by the XML Schema language. The vocabulary includes constructs for defining and ordering, default values, mandatory content, nesting, repeated sets, and redefines. Oracle XML DB supports all the constructs, except for redefines.

### Text Representation of the PurchaseOrder XML Schema

The following example `purchaseOrder.xsd`, is a standard W3C XML schema example fragment, in its native form, as an XML Document:

***Example 3–7 Purchase-Order XML Schema, purchaseOrder.xsd***

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0">
  <xs:element name="PurchaseOrder" type="PurchaseOrderType"/>
  <xs:complexType name="PurchaseOrderType">
    <xs:sequence>
```

```
    <xs:element name="Reference" type="ReferenceType"/>
    <xs:element name="Actions" type="ActionsType"/>
    <xs:element name="Reject" type="RejectionType" minOccurs="0"/>
    <xs:element name="Requestor" type="RequestorType"/>
    <xs:element name="User" type="UserType"/>
    <xs:element name="CostCenter" type="CostCenterType"/>
    <xs:element name="ShippingInstructions" type="ShippingInstructionsType"/>
    <xs:element name="SpecialInstructions" type="SpecialInstructionsType"/>
    <xs:element name="LineItems" type="LineItemsType"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="LineItemsType">
  <xs:sequence>
    <xs:element name="LineItem" type="LineItemType" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="LineItemType">
  <xs:sequence>
    <xs:element name="Description" type="DescriptionType"/>
    <xs:element name="Part" type="PartType"/>
  </xs:sequence>
  <xs:attribute name="ItemNumber" type="xs:integer"/>
</xs:complexType>
<xs:complexType name="PartType">
  <xs:attribute name="Id">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="10"/>
        <xs:maxLength value="14"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="Quantity" type="moneyType"/>
  <xs:attribute name="UnitPrice" type="quantityType"/>
</xs:complexType>
<xs:simpleType name="ReferenceType">
  <xs:restriction base="xs:string">
    <xs:minLength value="18"/>
    <xs:maxLength value="30"/>
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="ActionsType">
  <xs:sequence>
    <xs:element name="Action" maxOccurs="4">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="User" type="UserType"/>
          <xs:element name="Date" type="DateType" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="RejectionType">
  <xs:all>
    <xs:element name="User" type="UserType" minOccurs="0"/>
    <xs:element name="Date" type="DateType" minOccurs="0"/>
    <xs:element name="Comments" type="CommentsType" minOccurs="0"/>
  </xs:all>
</xs:complexType>
```

```
<xs:complexType name="ShippingInstructionsType">
  <xs:sequence>
    <xs:element name="name" type="NameType" minOccurs="0"/>
    <xs:element name="address" type="AddressType" minOccurs="0"/>
    <xs:element name="telephone" type="TelephoneType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:simpleType name="moneyType">
  <xs:restriction base="xs:decimal">
    <xs:fractionDigits value="2"/>
    <xs:totalDigits value="12"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="quantityType">
  <xs:restriction base="xs:decimal">
    <xs:fractionDigits value="4"/>
    <xs:totalDigits value="8"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="UserType">
  <xs:restriction base="xs:string">
    <xs:minLength value="0"/>
    <xs:maxLength value="10"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="RequestorType">
  <xs:restriction base="xs:string">
    <xs:minLength value="0"/>
    <xs:maxLength value="128"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="CostCenterType">
  <xs:restriction base="xs:string">
    <xs:minLength value="1"/>
    <xs:maxLength value="4"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="VendorType">
  <xs:restriction base="xs:string">
    <xs:minLength value="0"/>
    <xs:maxLength value="20"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="PurchaseOrderNumberType">
  <xs:restriction base="xs:integer"/>
</xs:simpleType>
<xs:simpleType name="SpecialInstructionsType">
  <xs:restriction base="xs:string">
    <xs:minLength value="0"/>
    <xs:maxLength value="2048"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="NameType">
  <xs:restriction base="xs:string">
    <xs:minLength value="1"/>
    <xs:maxLength value="20"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="AddressType">
  <xs:restriction base="xs:string">
```

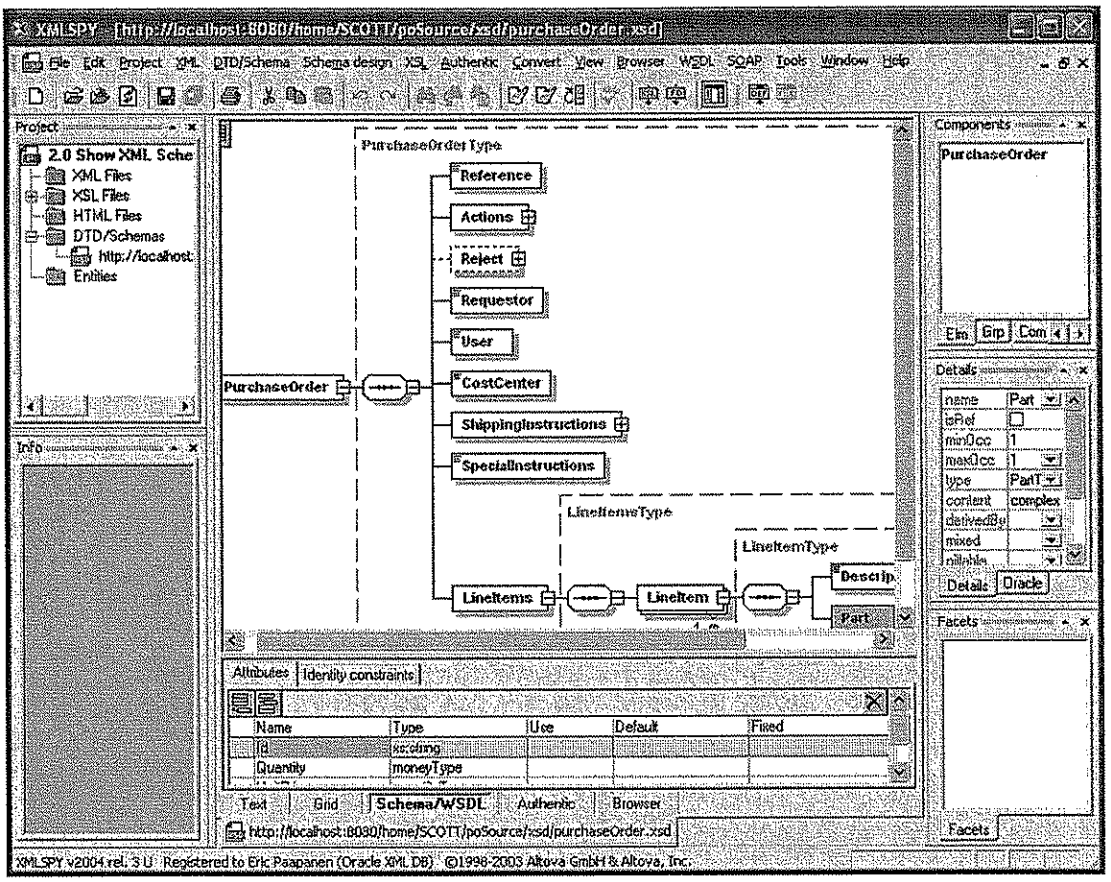
```
      <xs:minLength value="1"/>
      <xs:maxLength value="256"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="TelephoneType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="24"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="DateType">
    <xs:restriction base="xs:date"/>
  </xs:simpleType>
  <xs:simpleType name="CommentsType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="2048"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="DescriptionType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="256"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

**See Also:** Appendix A, "XML Schema Primer" for a more detailed listing of XML schema `purchaseOrder.xsd`

### Graphical Representation of the Purchase-Order XML Schema

Figure 3–2 shows the purchase-order XML schema displayed using XMLSpy. XMLSpy is a graphical and user-friendly tool from Altova Corporation for creating and editing XML schema and XML documents. See `http://www.altova.com` for details. XMLSpy also supports WebDAV and FTP protocols hence can directly access and edit content stored in Oracle XML DB Repository.

***Figure 3–2 XMLSpy Graphical Representation of the PurchaseOrder XML Schema***




The PurchaseOrder XML schema is a simple XML schema that demonstrates key features of a typical XML document:

- Global element `PurchaseOrder` is an instance of the `complexType` `PurchaseOrderType`
- `PurchaseOrderType` defines the set of nodes that make up a `PurchaseOrder` element
- `LineItems` element consists of a collection of `LineItem` elements
- Each `LineItem` element consists of two elements: `Description` and `Part`
- `Part` element has attributes `Id`, `Quantity`, and `UnitPrice`

# Using XML Schema with Oracle XML DB

This section describes the use of XML Schema with Oracle XML DB.

## Why Use XML Schema With Oracle XML DB?

The following paragraphs describe the main reasons for using XML schema with Oracle XML DB.

### Validating Instance Documents with XML Schema

The most common usage of XML Schema is as a mechanism for validating that instance documents conform to a given XML schema. The `XMLType` datatype methods `isSchemaValid()` and `schemaValidate()` allow Oracle XML DB to validate the contents of an instance document stored in an `XMLType`, against an XML schema.

### Constraining Instance Documents for Business Rules or Format Compliance

An XML schema can also be used as a constraint when creating tables or columns of `XMLType`. For example, the `XMLType` is constrained to storing XML documents compliant with one of the global elements defined by the XML schema.

### Defining How XMLType Contents Must be Stored in the Database

Oracle XML DB also uses XML schema as a mechanism for defining how the contents of an `XMLType` should be stored inside the database. Currently Oracle XML DB provides two options:

- Unstructured storage. The content of the `XMLType` is persisted as XML text using a `CLOB` datatype. This option is available for non-schema-based and schema-based XML content. When the XML is to be stored and retrieved as complete documents, unstructured storage may be the best solution as it offers the fastest rates of throughput when storing and retrieving XML content.
- Structured storage. The content of the `XMLType` is persisted as a set of SQL objects. The structured storage option is only available when the `XMLType` table or column has been constrained to a global element defined by XML schema.

  If there is a need to extract or update sections of the document, perform XSL transformation on the document, or work through the DOM API, then structured storage may be the preferred storage type. Structured storage allows all these operations to take place more efficiently but at a greater overhead when storing and retrieving the entire document.

## Structured Storage of XML Documents

Structured storage of XML documents is based on decomposing the content of the document into a set of SQL objects. These SQL objects are based on the SQL 1999 Type framework. When an XML schema is registered with Oracle XML DB, the required SQL type definitions are automatically generated from the XML schema.

A SQL type definition is generated from each `complexType` defined by the XML schema. Each element or attribute defined by the `complexType` becomes a SQL attribute in the corresponding SQL type. Oracle XML DB automatically maps the 47 scalar data types defined by the XML Schema Recommendation to the 19 scalar datatypes supported by SQL. A varray type is generated for each element and this can occur multiple times.

The generated SQL types allow XML content, compliant with the XML schema, to be decomposed and stored in the database as a set of objects without any loss of information. When the document is ingested the constructs defined by the XML schema are mapped directly to the equivalent SQL types.

This allows Oracle XML DB to leverage the full power of Oracle Database when managing XML and can lead to significant reductions in the amount of space required to store the document. It can also reduce the amount of memory required to query and update XML content.

## Annotating an XML Schema to Control Naming, Mapping, and Storage

The W3C XML Schema Recommendation defines an annotation mechanism that allows vendor-specific information to be added to an XML schema. Oracle XML DB uses this to control the mapping between the XML schema and the SQL object model.

Annotating an XML schema allows control over the naming of the SQL objects and attributes created. Annotations can also be used to override the default mapping between the XML schema data types and SQL data types and to specify which table should be used to store the data.

---

**Note:** As always:

- SQL is case-insensitive, but names in SQL code are *implicitly uppercase*, unless you enclose them in double-quotes.
- *XML is case-sensitive.* You must refer to SQL names in XML code using the correct case: uppercase SQL names must be written as uppercase.

For example, if you create a table named `my_table` in SQL without using double-quotes, then you must refer to it in XML as "`MY_TABLE`".

---

## Controlling How XML Collections are Stored in the Database

Annotations are also used to control how collections in an XML document are stored in the database. Currently there are four options:

- Character Large Object (`CLOB`). The entire set of elements is persisted as XML text stored in a `CLOB` column.
- Varray in a LOB. Each element in the collection is converted into a SQL object. The collection of SQL objects is serialized and stored in a LOB column.
- Varray as a nested table. Each element in the collection is converted into a SQL object. The collection of SQL objects is stored as a set of rows in an Index Organized Nested Table (IOT).
- Varray as a set of `XMLType` values. Each element in the collection is treated as a separate `XMLType` value. The collection of `XMLType` values is stored as a set of rows in an `XMLType` table.

These storage options allow you to tune the performance of applications that use `XMLType` datatypes to store XML in the database.

However, there is no requirement to annotate an XML schema before using it with Oracle XML DB. Oracle XML DB uses a set of default assumptions when processing an XML schema that contains no annotations.

**See Also:** Chapter 5, "XML Schema Storage and Query: Basic"

### Collections: Default Mapping

When no annotations are supplied by the user, Oracle XML DB stores a collection as a varray in a LOB.

## Declaring the Oracle XML DB Namespace

Before annotating an XML schema you must first declare the Oracle XML DB namespace. The Oracle XML DB namespace is defined as:

```
http://xmlns.oracle.com/xdb
```

The namespace is declared in the XML schema by adding a namespace declaration such as the following to the root element of the XML schema:

```
xmlns:xdb="http://xmlns.oracle.com/xdb"
```

Note the use of a namespace prefix (xdb). This makes it possible to abbreviate the namespace to xdb when adding annotations.

Example 3–8 shows the beginning of the `PurchaseOrder` XML schema with annotations. See Example D–1 on page D-13 for the complete schema listing.

***Example 3–8 Annotated Purchase-Order XML Schema, purchaseOrder.xsd***

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:xdb="http://xmlns.oracle.com/xdb"
           version="1.0"
           xdb:storeVarrayAsTable="true">
  <xs:element name="PurchaseOrder" type="PurchaseOrderType" xdb:defaultTable="PURCHASEORDER"/>
  <xs:complexType name="PurchaseOrderType" xdb:SQLType="PURCHASEORDER_T">
    <xs:sequence>
      <xs:element name="Reference" type="ReferenceType" minOccurs="1" xdb:SQLName="REFERENCE"/>
      <xs:element name="Actions" type="ActionsType" xdb:SQLName="ACTIONS"/>
      <xs:element name="Reject" type="RejectionType" minOccurs="0" xdb:SQLName="REJECTION"/>
      <xs:element name="Requestor" type="RequestorType" xdb:SQLName="REQUESTOR"/>
      <xs:element name="User" type="UserType" minOccurs="1" xdb:SQLName="USERID"/>
      <xs:element name="CostCenter" type="CostCenterType" xdb:SQLName="COST_CENTER"/>
      <xs:element name="ShippingInstructions" type="ShippingInstructionsType"
                  xdb:SQLName="SHIPPING_INSTRUCTIONS"/>
      <xs:element name="SpecialInstructions" type="SpecialInstructionsType"
                  xdb:SQLName="SPECIAL_INSTRUCTIONS"/>
      <xs:element name="LineItems" type="LineItemsType" xdb:SQLName="LINEITEMS"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="LineItemsType" xdb:SQLType="LINEITEMS_T">
    <xs:sequence>
      <xs:element name="LineItem" type="LineItemType" maxOccurs="unbounded"
                  xdb:SQLName="LINEITEM" xdb:SQLCollType="LINEITEM_V"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="LineItemType" xdb:SQLType="LINEITEM_T">
    <xs:sequence>
      <xs:element name="Description" type="DescriptionType"
                  xdb:SQLName="DESCRIPTION"/>
      <xs:element name="Part" type="PartType" xdb:SQLName="PART"/>
    </xs:sequence>
    <xs:attribute name="ItemNumber" type="xs:integer" xdb:SQLName="ITEMNUMBER"
                  xdb:SQLType="NUMBER"/>
  </xs:complexType>
  <xs:complexType name="PartType" xdb:SQLType="PART_T">
    <xs:attribute name="Id" xdb:SQLName="PART_NUMBER" xdb:SQLType="VARCHAR2">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:minLength value="10"/>
          <xs:maxLength value="14"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Quantity" type="moneyType" xdb:SQLName="QUANTITY"/>
    <xs:attribute name="UnitPrice" type="quantityType" xdb:SQLName="UNITPRICE"/>
  </xs:complexType>
  <xs:simpleType name="ReferenceType">
    <xs:restriction base="xs:string">
      <xs:minLength value="18"/>
      <xs:maxLength value="30"/>
```

```
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="ActionsType" xdb:SQLType="ACTIONS_T">
  <xs:sequence>
    <xs:element name="Action" maxOccurs="4" xdb:SQLName="ACTION" xdb:SQLCollType="ACTION_V">
      <xs:complexType xdb:SQLType="ACTION_T">
        <xs:sequence>
          <xs:element name="User" type="UserType" xdb:SQLName="ACTIONED_BY"/>
          <xs:element name="Date" type="DateType" minOccurs="0" xdb:SQLName="DATE_ACTIONED"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="RejectionType" xdb:SQLType="REJECTION_T">
  <xs:all>
    <xs:element name="User" type="UserType" minOccurs="0" xdb:SQLName="REJECTED_BY"/>
    <xs:element name="Date" type="DateType" minOccurs="0" xdb:SQLName="DATE_REJECTED"/>
    <xs:element name="Comments" type="CommentsType" minOccurs="0" xdb:SQLName="REASON_REJECTED"/>
  </xs:all>
</xs:complexType>
<xs:complexType name="ShippingInstructionsType" xdb:SQLType="SHIPPING_INSTRUCTIONS_T">
  <xs:sequence>
    <xs:element name="name" type="NameType" minOccurs="0" xdb:SQLName="SHIP_TO_NAME"/>
    <xs:element name="address" type="AddressType" minOccurs="0" xdb:SQLName="SHIP_TO_ADDRESS"/>
    <xs:element name="telephone" type="TelephoneType" minOccurs="0" xdb:SQLName="SHIP_TO_PHONE"/>
  </xs:sequence>
</xs:complexType>
<xs:simpleType name="moneyType">
  <xs:restriction base="xs:decimal">
    <xs:fractionDigits value="2"/>
    <xs:totalDigits value="12"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="quantityType">
  <xs:restriction base="xs:decimal">
    <xs:fractionDigits value="4"/>
    <xs:totalDigits value="8"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="UserType">
  <xs:restriction base="xs:string">
    <xs:minLength value="0"/>
    <xs:maxLength value="10"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="RequestorType">
  <xs:restriction base="xs:string">
    <xs:minLength value="0"/>
    <xs:maxLength value="128"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="CostCenterType">
  <xs:restriction base="xs:string">
    <xs:minLength value="1"/>
    <xs:maxLength value="4"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="VendorType">
  <xs:restriction base="xs:string">
    <xs:minLength value="0"/>
    <xs:maxLength value="20"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="PurchaseOrderNumberType">
  <xs:restriction base="xs:integer"/>
```

```
  </xs:simpleType>
  <xs:simpleType name="SpecialInstructionsType">
    <xs:restriction base="xs:string">
      <xs:minLength value="0"/>
      <xs:maxLength value="2048"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="NameType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="20"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="AddressType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="256"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="TelephoneType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="24"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="DateType">
    <xs:restriction base="xs:date"/>
  </xs:simpleType>
  <xs:simpleType name="CommentsType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="2048"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="DescriptionType">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="256"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

The `PurchaseOrder` XML schema defines the following two *namespaces*:

- `http://www.w3c.org/2001/XMLSchema`. This is reserved by W3C for the Schema for Schemas.
- `http://xmlns.oracle.com/xdb`. This is reserved by Oracle for the Oracle XML DB schema annotations.

The `PurchaseOrder` schema uses several *annotations*, including the following:

- `defaultTable` annotation in the `PurchaseOrder` element. This specifies that XML documents, compliant with this XML schema are stored in a database table called `purchaseorder`.
- `SQLType` annotation.

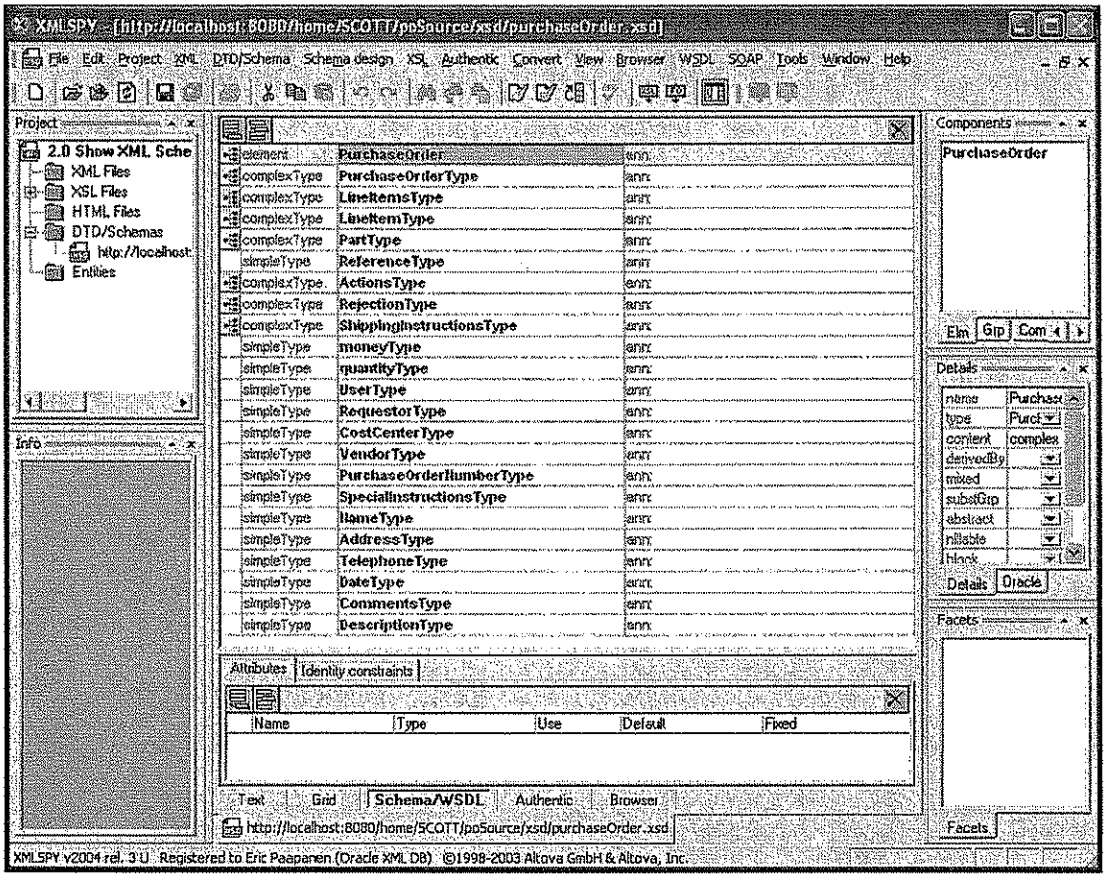
  The first occurrence of `SQLType` specifies that the name of the SQL type generated from complexType `PurchaseOrderType` is `purchaseorder_t`.

  The second occurrence of `SQLType` specifies that the name of the SQL type generated from the complexType `LineItemType` is `lineitem_t` and the SQL type that manages the collection of `LineItem` elements is `lineitem_v`.

- SQLName annotation. This provides an explicit name for each SQL attribute of purchaseorder_t.

Figure 3–3 shows the XMLSpy Oracle tab, which facilitates adding Oracle XML DB schema annotations to an XML schema while working in the graphical editor.

***Figure 3–3 XMLSpy Showing Support for Oracle XML DB Schema Annotations***




## Registering an XML Schema with Oracle XML DB

For an XML schema to be useful to Oracle XML DB you must first register it with Oracle XML DB. After it has been registered, it can be used for validating XML documents and for creating XMLType tables and columns bound to the XML schema.

Two items are required to register an XML schema with Oracle XML DB:

- The XML schema document
- A string that can be used as a unique identifier for the XML schema, after it is registered with Oracle Database. Instance documents use this unique identifier to identify themselves as members of the class defined by the XML schema. The identifier is typically in the form of a URL, and is often referred to as the **schema location hint**.

You register an XML schema with PL/SQL procedure DBMS_XMLSCHEMA.registerschema. See Example 3–9. By default, when an XML schema is registered, Oracle XML DB automatically generates all of the SQL object types and XMLType tables required to manage the instance documents.

XML schemas can be registered as global or local.

***Example 3–9 Registering an XML Schema with DBMS_XMLSCHEMA.registerSchema***

```
BEGIN
  DBMS_XMLSCHEMA.registerSchema(
    'http://localhost:8080/source/schemas/poSource/xsd/purchaseOrder.xsd',
    XDBURIType('/so
```

# Exhibit 13

# Oracle® Real Application Clusters

Administration and Deployment Guide

11*g* Release 1 (11.1)

**B28254-04**

November 2007

ORACLE

Oracle Real Application Clusters Administration and Deployment Guide, 11g Release 1 (11.1)

B28254-04

Copyright © 1999, 2007, Oracle. All rights reserved.

Primary Authors: Mark Bauer, Richard Strohm

Contributing Authors: Troy Anthony, Lance Ashdown, Prasad Bagal, Anand Beldalker, Eric Belden, George Claborn, Carol Colrain, Jonathan Creighton, Rajesh Dasari, Steve Fogel, Richard Frank, GP Prabhaker Gongloor, Wei Hu, Yong Hu, Rajiv Jayaraman, Dominique Jeunot, Sameer Joshi, Roland Knapp, Raj Kumar, Ken Lee, Karen Li, Barb Lundhild, Venkat Maddali, Bill Manry, Gaurav Manglik, Komal Mangtani, John McHugh, Saar Maoz, Matthew Mckerley, Markus Michalewicz, Anil Nair, Philip Newlan, Michael Nowak, Peter Ogilvie, Muthu Olagappan, Bharat Paliwal, Hanlin Qian, Mark Ramacher, Kevin Reardon, Dipak Saggi, Sudheendra Sampath, Viv Schupmann, Daniel Semler, Ara Shakian, Cathy Shea, Khethavath P. Singh, Bipul Sinha, Qun Song, Kesavan Srinivasan, Alok Srivastava, Janet Stern, Juan Tellez, Leo Tominna, Peter Wahl, Tak Wang, Douglas Williams, Mike Zampiceni, Michael Zoll

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

# 1

# Introduction to Oracle Real Application Clusters

This chapter introduces Oracle Real Application Clusters (Oracle RAC) and describes how to install, administer, and deploy Oracle RAC.

This chapter includes the following topics:

- Overview of Oracle Real Application Clusters
- Overview of Oracle Clusterware for Oracle Real Application Clusters
- Overview of Oracle Real Application Clusters Architecture and Processing
- Overview of Automatic Workload Management
- Overview of Installing Oracle Real Application Clusters
- Overview of Managing Oracle Real Application Clusters Environments

## Overview of Oracle Real Application Clusters

A **cluster** comprises multiple interconnected computers or servers that appear as if they are one server to end users and applications. Oracle Real Application Clusters (Oracle RAC) enables you to cluster Oracle databases. Oracle RAC uses Oracle Clusterware for the infrastructure to bind multiple servers so they operate as a single system.

Oracle Clusterware is a portable cluster management solution that is integrated with the Oracle database. Oracle Clusterware is also a required component for using Oracle RAC. In addition, Oracle Clusterware enables both single-instance Oracle databases and Oracle RAC databases to use the Oracle high-availability infrastructure. Oracle Clusterware enables you to create a clustered pool of storage to be used by any combination of single-instance and Oracle RAC databases.

Oracle Clusterware is the only clusterware that you need for most platforms on which Oracle RAC operates. You can also use clusterware from other vendors if the clusterware is certified for Oracle RAC.

> **See Also:** *Oracle Clusterware Administration and Deployment Guide* and the Oracle Clusterware install guides for more details

Single-instance Oracle databases have a one-to-one relationship between the Oracle database and the instance. Oracle RAC environments, however, have a one-to-many relationship between the database and instances. An Oracle RAC database can have up to 100 instances,[1] all of which access one database. All database instances must use the same interconnect, which can also be used by Oracle Clusterware.

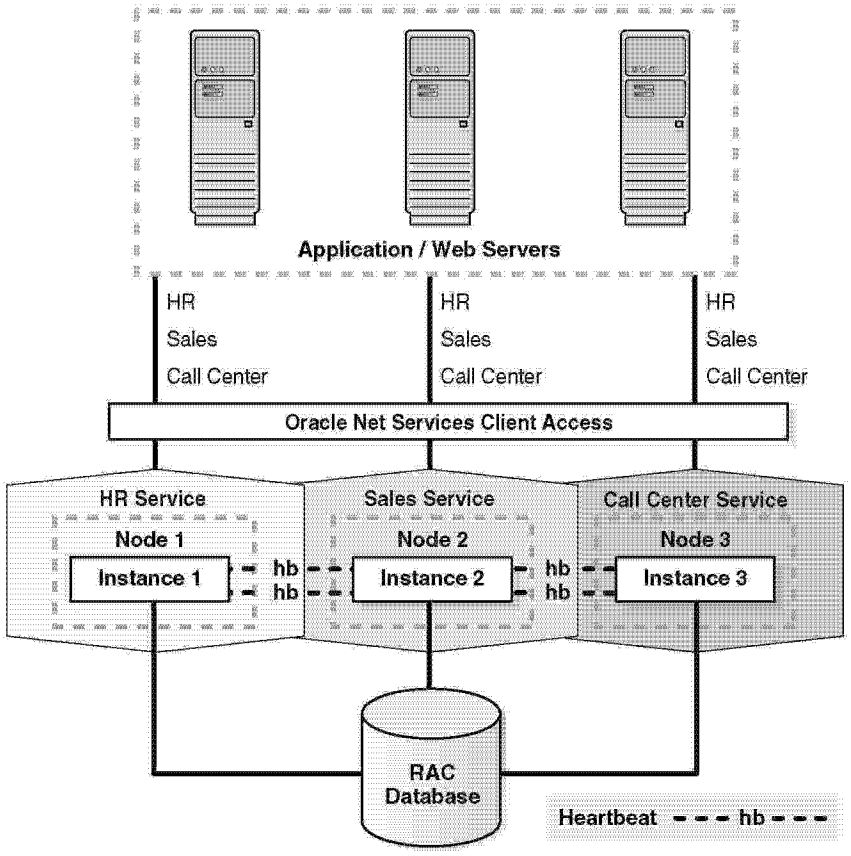
Oracle RAC databases differ architecturally from single-instance Oracle databases in that each Oracle RAC database instance also has:

- At least one additional thread of redo for each instance
- An instance-specific undo tablespace

The combined processing power of the multiple servers can provide greater throughput and Oracle RAC scalability than is available from a single server.

Figure 1–1 shows how Oracle RAC is the Oracle Database option that provides a single system image for multiple servers to access one Oracle database. In Oracle RAC, each Oracle instance usually runs on a separate server.

***Figure 1–1 Oracle Database with Oracle RAC Architecture***




Traditionally, an Oracle RAC environment is located in one data center. However, you can configure Oracle RAC on an extended distance cluster, which is an architecture that provides extremely fast recovery from a site failure and allows for all nodes, at all sites, to actively process transactions as part of single database cluster. In an extended distance cluster, the nodes in the cluster are located in two buildings that are separated by greater distances (anywhere from across the street, to across a campus, or across a city). For availability reasons, the data needs to be located at both sites, thus requiring the need to implement disk mirroring technology for storage.

If you choose to implement this architecture, you must assess whether this architecture is a good solution for your business, especially with regard to distance, latency, and

[1] For configurations running Oracle Database 10g Release 2 and later releases, Oracle Clusterware supports 100 nodes in a cluster, and Oracle RAC supports 100 instances in an Oracle RAC database.



ORCL01972354

the degree of protection it provides. Oracle RAC on extended distance clusters provides higher availability than is possible with a local Oracle RAC configurations, but an extended distance cluster may not fulfill all of the disaster-recovery requirements of your organization. A feasible separation provides great protection for some disasters (for example, local power outage, airplane crash, server room flooding) but it cannot provide protection against all types of outages. For comprehensive protection against disasters—including protection against corruptions and regional disasters—Oracle recommends the use of Oracle Data Guard with Oracle RAC, as described in the *Oracle Database High Availability Overview* and on the Maximum Availability Architecture (MAA) Web site at

`http://www.oracle.com/technology/deploy/availability/htdocs/maa.htm`

Oracle RAC is a unique technology that provides high availability and scalability for all application types. The Oracle RAC infrastructure is also a key component for implementing the Oracle enterprise grid computing architecture. Having multiple instances access a single database prevents the server from being a single point of failure. Oracle RAC enables you to combine smaller commodity servers into a cluster to create scalable environments that support mission critical business applications. Applications that you deploy on Oracle RAC databases can operate without code changes.

## Overview of Oracle Clusterware for Oracle Real Application Clusters

Oracle Clusterware provides a complete, integrated clusterware management solution on all Oracle Database platforms. This clusterware functionality provides all of the features required to manage your cluster database including node membership, group services, global resource management, and high availability functions.

You can install Oracle Clusterware independently or as a prerequisite to the Oracle RAC installation process. Oracle database features such as services use the underlying Oracle Clusterware mechanisms to provide their capabilities. Oracle also continues to support select third-party clusterware products on specified platforms.

Oracle Clusterware is designed for, and tightly integrated with, Oracle RAC. When you create an Oracle RAC database using any of the management tools, the database is registered with and managed by Oracle Clusterware, along with the other Oracle processes such as Virtual Internet Protocol (VIP) address, Global Services Daemon (GSD), the Oracle Notification Service (ONS), and the Oracle Net listeners. These resources are automatically started when Oracle Clusterware starts the node and automatically restarted if they fail. The Oracle Clusterware daemons run on each node.

You can use Oracle Clusterware to manage high-availability operations in a cluster. Anything that Oracle Clusterware manages is known as a CRS resource, which could be a database, an instance, a service, a listener, a VIP address, an application process, and so on. Oracle Clusterware manages CRS resources based on the resource's configuration information that is stored in the Oracle Cluster Registry (OCR). You can use `SRVCTL` commands to administer other node resources. Oracle Clusterware stores the information that describes the configuration of these components in the OCR that you can administer as described in the *Oracle Clusterware Administration and Deployment Guide*.

# Overview of Oracle Real Application Clusters Architecture and Processing

At a minimum, Oracle RAC requires a cluster software infrastructure that can provide concurrent access to the same storage and the same set of data files from all nodes in the cluster, a communications protocol for enabling interprocess communication (IPC) across the nodes in the cluster, enable multiple database instances to process data as if the data resided on a logically combined, single cache, and a mechanism for monitoring and communicating the status of the nodes in the cluster.

The following sections describe these concepts in more detail:

- Understanding Cluster-Aware Storage Solutions
- Overview of Connecting to the Oracle Database Using Services and VIP Addresses
- About Oracle Real Application Clusters Software Components
- About Oracle Real Application Clusters Background Processes

## Understanding Cluster-Aware Storage Solutions

An Oracle RAC database is a **shared everything** database. All data files, control files, SPFILEs,[2] and redo log files in Oracle RAC environments must reside on cluster-aware shared disks so that all of the cluster database instances can access these storage components. All database instances must use the same interconnect, which can also be used by Oracle Clusterware. Because Oracle RAC databases use a shared everything architecture, Oracle RAC requires cluster-aware storage for all database files.

In Oracle RAC, the Oracle Database software manages disk access and the Oracle software is certified for use on a variety of storage architectures. It is your choice as to how to configure your disk, but you must use a supported cluster-aware storage solution. Oracle Database provides the following file storage options for Oracle RAC:

- Automatic Storage Management (ASM)

  This is the recommended solution to manage your disk.
- OCFS2 and Oracle Cluster File System (OCFS)

  OCFS2 is available for Linux and OCFS is available for Windows platforms. However you may optionally use a third-party cluster file system or cluster-aware volume manager that is certified for Oracle RAC.
- A network file system
- Raw devices

## Overview of Connecting to the Oracle Database Using Services and VIP Addresses

All **nodes** in an Oracle RAC environment must connect to a Local Area Network (LAN) to enable users and applications to access the database. Applications should use the Oracle Database services feature to connect to an Oracle database. Services enable you to define rules and characteristics to control how users and applications connect to database instances. These characteristics include a unique name, workload balancing and failover options, and high availability characteristics. Oracle Net Services enable the load balancing of application connections across all of the instances in an Oracle RAC database.

[2] Note that PFILE files do not need to be shared.



ORCL01972356

Users can access an Oracle RAC database using a client/server configuration or through one or more middle tiers, with or without connection pooling. Users can be database administrators, developers, application users, power users, such as data miners who create their own searches, and so on.

Most public networks typically use TCP/IP, but you can use any supported hardware and software combination. Oracle RAC database instances can be accessed through a database's default IP address and through VIP addresses.

The interconnect network is a private network that connects all of the servers in the cluster. The interconnect network uses a switch (or multiple switches) that only the nodes in the cluster can access. Configure User Datagram Protocol (UDP) on a Gigabit Ethernet for your cluster interconnect. On Linux and Unix systems, you can configure Oracle Clusterware to use either the UDP or Reliable Data Socket (RDS) protocols. Windows clusters use the TCP protocol. Crossover cables are not supported for use with Oracle Clusterware interconnects.

---

**Note:** Do not use the interconnect for the private network for user communication, because **Cache Fusion** uses the private interconnect for interinstance communication.

---

In addition to the node's host name and IP address, you must also assign a virtual host name and an IP address to each node. You should use the virtual host name or VIP address to connect to the database instance. For example, you might enter the virtual host name `CRM` in the address list of the `tnsnames.ora` file.

A virtual IP address is an alternate public address that client connections use instead of the standard public IP address. To configure VIP addresses, you need to reserve a spare IP address for each node, and the IP addresses must use the same subnet as the public network.

If a node fails, then the node's VIP address fails over to another node on which the VIP address can accept TCP connections but it cannot accept Oracle connections. Generally, VIP addresses fail over when:

- The node on which a VIP address runs fails
- All interfaces for the VIP address fail
- All interfaces for the VIP address are disconnected from the network

Clients that attempt to connect to the VIP address receive a rapid `connection refused` error instead of waiting for TCP connect timeout messages. You configure VIP addresses in the address list for your database connection definition to enable connectivity.

If you use Network Attached Storage (NAS), then you are required to configure a second private network. Access to this network is typically controlled by the vendor's software. The private network uses static IP addresses.

## About Oracle Real Application Clusters Software Components

Oracle RAC databases have two or more database instances that each contain memory structures and background processes. An Oracle RAC database has the same processes and memory structures as a single-instance Oracle database as well as additional process and memory structures that are specific to Oracle RAC. Any one instance's database view is nearly identical to any other instance's view in the same Oracle RAC database; the view is a single system image of the environment.



ORCL01972357

Each instance has a buffer cache in its System Global Area (SGA). Using Cache Fusion, Oracle RAC environments logically combine each instance's buffer cache to enable the instances to process data as if the data resided on a logically combined, single cache.

> **Note:** The SGA size requirements for Oracle RAC are greater than the SGA requirements for single-instance Oracle databases due to Cache Fusion.

To ensure that each Oracle RAC database instance obtains the block that it needs to satisfy a query or transaction, Oracle RAC instances use two processes, the Global Cache Service (GCS) and the Global Enqueue Service (GES). The GCS and GES maintain records of the statuses of each data file and each cached block using a Global Resource Directory (GRD). The GRD contents are distributed across all of the active instances, which effectively increases the size of the SGA for an Oracle RAC instance.

After one instance caches data, any other instance within the same cluster database can acquire a block image from another instance in the same database faster than by reading the block from disk. Therefore, Cache Fusion moves current blocks between instances rather than re-reading the blocks from disk. When a consistent block is needed or a changed block is required on another instance, Cache Fusion transfers the block image directly between the affected instances. Oracle RAC uses the private interconnect for interinstance communication and block transfers. The GES Monitor and the Instance Enqueue Process manages access to Cache Fusion resources and enqueue recovery processing.

## About Oracle Real Application Clusters Background Processes

The GCS and GES processes, and the GRD collaborate to enable Cache Fusion. The Oracle RAC processes and their identifiers are as follows:

- `ACMS`—Atomic Controlfile to Memory Service (ACMS)

  In an Oracle RAC environment, the atomic controlfile to memory service (ACMS) per-instance process is an agent that contributes to ensuring a distributed SGA memory update is either globally committed on success or globally aborted in the event of a failure.

- `GTX0-j`—Global Transaction Process

  The GTX0-j process provides transparent support for XA global transactions in a RAC environment. The database autotunes the number of these processes based on the workload of XA global transactions.

- `LMON`—Global Enqueue Service Monitor

  The LMON process monitors global enqueues and resources across the cluster and performs global enqueue recovery operations.

- `LMD`—Global Enqueue Service Daemon

  The LMD process manages incoming remote resource requests within each instance.

- `LMS`—Global Cache Service Process

  The LMS process maintains records of the datafile statuses and each cached block by recording information in a Global Resource Directory (GRD). The LMS process also controls the flow of messages to remote instances and manages global data



ORCL01972358

block access and transmits block images between the buffer caches of different instances. This processing is part of the Cache Fusion feature.

- LCK0—Instance Enqueue Process

  The LCK0 process manages non-Cache Fusion resource requests such as library and row cache requests.

- RMS*n*—Oracle RAC Management Processes (RMS*n*)

  The RMS*n* processes perform manageability tasks for Oracle RAC. Tasks accomplished by an RMS*n* process include creation of resources related Oracle RAC when new instances are added to the clusters.

- RSMN—Remote Slave Monitor manages background slave process creation and communication on remote instances. These background slave processes perform tasks on behalf of a coordinating process running in another instance.

**Note:** Many of the Oracle components that this section describes are in addition to the components that are described for single-instance Oracle databases in *Oracle Database Concepts*.

## Overview of Automatic Workload Management

Automatic workload management enables you to manage the distribution of workloads to provide optimal performance for users and applications. This includes providing the highest availability for database connections, rapid failure recovery, and balancing workloads optimally across the active configuration. Oracle Database with Oracle RAC includes many features that can enhance automatic workload management, such as connection load balancing, fast connection failover, the load balancing advisory, and runtime connection load balancing. Automatic workload management provides the greatest benefits to Oracle RAC environments. You can, however, take advantage of automatic workload management by using Oracle services in single-instance Oracle Databases, especially those that use Data Guard or Streams. Automatic workload management comprises the following components:

- **High Availability Framework**—The Oracle RAC high availability framework enables the Oracle Database to maintain components in a running state at all times. Oracle high availability implies that Oracle Clusterware monitors and restarts critical components if they stop, unless you override the restart processing. Oracle Clusterware and Oracle RAC also provide alerts to clients when configurations change. This enables clients to immediately react to the changes, enabling application developers to hide outages and reconfigurations from end users. The scope of Oracle high availability spans from the restarting of stopped Oracle processes in an Oracle database instance to failing over the processing of an entire instance to other available instances.

- **Load Balancing Advisory**—This is the ability of the database to provide information to applications about the current service levels being provided by the database and its instances. Applications can take advantage of this information to direct connection requests to the instance that will provide the application request with the best service quality to complete the application's processing. Oracle has integrated its Java Database Connectivity (JDBC) and Oracle Data Provider for .NET (ODP.NET) connection pools to work with the load balancing information. Applications can use the integrated connection pools without programmatic changes.



ORCL01972359

- **Services**—Oracle Database provides a powerful automatic workload management facility, called services, to enable the enterprise grid vision. Services are entities that you can define in Oracle RAC databases. Services enable you to group database workloads and route the work to the optimal instances that are assigned to process the service. Furthermore, you can use services to define the resources that Oracle assigns to process workloads and to monitor workload resources. Applications that you assign to services transparently acquire the defined automatic workload management characteristics, including high availability and load balancing rules. Many Oracle database features are integrated with services, such as Resource Manager, which enables you to restrict the resources that a service can use within an instance. Some database features are also integrated with Oracle Streams, Advanced Queuing (to achieve queue location transparency), and Oracle Scheduler (to map services to specific job classes).

  In Oracle RAC databases, the service performance rules that you configure control the amount of work that Oracle allocates to each available instance for that service. As you extend your database by adding nodes, applications, components of applications, and so on, you can add more services.

- **Connection Load Balancing**— Oracle Net Services provides connection load balancing for database connections. Connection load balancing occurs when the connection is created. Connections for a given service are balanced across all of the running instances that offer the service. You should define how you want connections to be balanced in the service definition. However, you must still configure Oracle Net Services. When you enable the load balancing advisory, the listener uses the load balancing advisory for connection load balancing.

  **See Also:** Chapter 4, "Introduction to Automatic Workload Management"

## Overview of Installing Oracle Real Application Clusters

Install Oracle Clusterware and Oracle Database software using Oracle Universal Installer (OUI), and create your database with Database Configuration Assistant (DBCA). This ensures that your Oracle RAC environment has the optimal network configuration, database structure, and parameter settings for the environment that you selected. As a database administrator, after installation your tasks are to administer your Oracle RAC environment at three levels:

- Instance Administration
- Database Administration
- Cluster Administration

This section introduces the installation processes for Oracle RAC under the following topics:

- Understanding Compatibility in Oracle Real Application Clusters Environments
- Overview of Oracle Real Application Clusters Installation and Database Creation
- Overview of Extending Oracle ASM and Oracle Real Application Clusters Software

---

**Note:** You must first install Oracle Clusterware before installing Oracle RAC. See *Oracle Clusterware Administration and Deployment Guide* for more information.

---



ORCL01972360

## Understanding Compatibility in Oracle Real Application Clusters Environments

To run Oracle RAC in configurations with different versions of the database in the same cluster, you must also install clusterware. For example, to run Oracle9*i* and Oracle 10*g* in the same cluster:

- For Oracle RAC nodes running the Oracle9*i* database, you must install an Oracle9*i* cluster:
  - For UNIX the cluster can be HACMP, Serviceguard, Sun Cluster, or Veritas SF
  - For Windows and Linux, the cluster is Oracle Cluster Manager
- If you want to install Oracle RAC running Oracle Database 10*g* or later releases, you must also install Oracle Clusterware. See your platform-specific Oracle Clusterware Installation guide and the *Oracle Clusterware Administration and Deployment Guide* for more information.
- If you are running Oracle RAC 10*g* and Oracle RAC 11*g* in the same cluster, you must be running Oracle Clusterware 11*g* (only)

Oracle requires that you install the Oracle9*i* RAC software first if you are going to run it in a cluster with Oracle RAC 10*g* or Oracle RAC 11*g*.

> **Note:** If you are adding Oracle RAC to servers that are part of a cluster, either migrate to Oracle Clusterware or ensure the clusterware you are running is supported to run with Oracle RAC on release 10*g* or later releases and ensure you have installed the correct options for the two to work together. Oracle strongly recommends that you do not run different cluster software on the same servers unless they are certified to work together.

## Overview of Oracle Real Application Clusters Installation and Database Creation

Once you have installed Oracle Clusterware and it is operational, run OUI to install the Oracle database software with Oracle RAC components.

During the installation, OUI runs DBCA to create your Oracle RAC database according to the options that you select. DBCA also runs the Net Configuration Assistant (NETCA) to configure the network for your Oracle RAC environment.

> **See Also:** *Oracle Database Net Services Administrator's Guide* for more information about NETCA

Oracle RAC software is distributed as part of the Oracle Database installation media. By default, the standard Oracle Database software installation process installs the Oracle RAC option when it recognizes that you are performing the installation on a cluster. The OUI installs Oracle RAC into a directory structure, which can be referred to as the Oracle home, which is separate from other Oracle software running on the system. Because OUI is cluster aware, it installs Oracle RAC software on all of the nodes that you defined to be part of the cluster.

Oracle recommends that you select ASM during the installation to simplify storage management; ASM automatically manages the storage of all database files within disk groups. You can also configure services during installation, depending on your processing requirements. If you are using Oracle Database Standard Edition, then you *must* use ASM to store all of the database files.



ORCL01972361

By default, Oracle creates one service for your environment and the service is for the database. (The default database service is typically identified using the combination of the `DB_NAME` and `DB_DOMAIN` initialization parameters: *`db_name.db_domain`*.) The default service is available on all instances in an Oracle RAC environment, unless the database is in restricted mode.

> **Note:** Avoid changing host names after you complete the Oracle Clusterware installation, including adding or deleting domain qualifications. Nodes with changed host names must be deleted from the cluster and added back with the new name.

## Overview of Extending Oracle ASM and Oracle Real Application Clusters Software

You can extend Oracle ASM and Oracle RAC in grid environments to additional nodes by copying cloned images of the ASM and Oracle RAC database homes to other nodes in the cluster. Oracle cloning copies images of the software to other nodes that have similar hardware and software. Cloning is best suited to a scenario where you need to quickly extend your Oracle RAC environment to several nodes of the same configuration.

Oracle provides the following methods of extending Oracle Clusterware environments:

- Oracle cloning procedure using cloning scripts
- Oracle Enterprise Manager cloning
- The `addNode.sh` script and OUI cloning

> **Note:** Oracle cloning is not a replacement for Oracle Enterprise Manager cloning that is part of the Provisioning Pack. During Oracle Enterprise Manager cloning, the provisioning process includes a series of steps where details about the home you want to capture, the location you want to deploy to, and various other parameters are collected.

For new installations or if you have to install only one Oracle RAC database, you should use the traditional automated and interactive installation methods, such as OUI, or the Provisioning Pack feature of Oracle Enterprise Manager. If your goal is to add or delete Oracle RAC from nodes in the cluster, you can use the `addNode.sh` and `rootdelete.sh` scripts.

The cloning process assumes that you successfully installed an Oracle Clusterware home and an Oracle home with Oracle RAC on at least one node. In addition, all root scripts must have run successfully on the node from which you are extending your cluster database.

At a high level, Oracle cloning involves the following main tasks:

1. Clone the Oracle Clusterware home following the instructions in *Oracle Clusterware Administration and Deployment Guide*.
2. Clone the Oracle home with the Oracle RAC software.

   The process for cloning the Oracle home onto new nodes is similar to the process for cloning the Oracle Clusterware home.
3. Run NETCA on each new node to create a listener.



ORCL01972362

4. If you have not already created a database, then run the DBCA to create one.
5. Follow the post-cloning procedures to complete the extension of your Oracle RAC environment onto the new nodes.

**See Also:**

- Chapter 7, "Using Cloning to Add ASM and Oracle RAC to Nodes in a New Cluster"
- Chapter 9, "Adding and Deleting Oracle RAC from Nodes on Linux and UNIX Systems" for information about adding and deleting nodes and instances on Linux and UNIX Systems
- Oracle Enterprise Manager online Help system for more information about the Provisioning Pack
- *Oracle Database Net Services Administrator's Guide* for more information about NETCA

# Overview of Managing Oracle Real Application Clusters Environments

This section describes the following Oracle RAC environment management topics:

- About Designing and Deploying Oracle Real Application Clusters Environments
- About Administrative Tools for Oracle Real Application Clusters Environments
- About Monitoring Oracle Real Application Clusters Environments
- About Evaluating Performance in Oracle Real Application Clusters Environments

## About Designing and Deploying Oracle Real Application Clusters Environments

Any enterprise that is designing and implementing a high availability strategy with Oracle RAC must begin by performing a thorough analysis of the business drivers that require high availability. An analysis of business requirements for high availability combined with an understanding of the level of investment required to implement different high availability solutions enables the development of a high availability architecture that will achieve both business and technical objectives. See the following resources for help choosing and implementing the architecture that best fits your availability requirements:

- Chapter 11, "Design and Deployment Techniques" provides a high-level overview you can use to evaluate the high availability requirements of your business.
- *Oracle Database High Availability Overview* describes how to select the most suitable architecture for your organization, describes several high availability architectures, and provides guidelines for choosing the one that best meets your requirements.

## About Administrative Tools for Oracle Real Application Clusters Environments

Oracle enables you to administer a cluster database as a single system image through Enterprise Manager, SQL*Plus, or through Oracle RAC command-line interfaces such as Server Control (`SRVCTL`):

- Oracle Enterprise Manager—Enterprise Manager has both the Database Control and Grid Control GUI interfaces for managing both single-instance database and Oracle RAC database environments. Oracle recommends that you use Enterprise Manager to perform administrative tasks whenever feasible.



ORCL01972363

- Server Control (SRVCTL)—SRVCTL is a command-line interface that you can use to manage an Oracle RAC database from a single point. You can use SRVCTL to start and stop the database and instances and to delete or move instances and services. You can also use SRVCTL to manage configuration information, Oracle Clusterware, and ASM.
- SQL*Plus—SQL*Plus commands operate on the current instance. The current instance can be either the local default instance on which you initiated your SQL*Plus session, or it can be a remote instance to which you connect with Oracle Net Services.
- Cluster Verification Utility (CVU)—CVU is a command-line tool that you can use to verify a range of cluster and Oracle RAC components such as shared storage devices, networking configurations, system requirements, and Oracle Clusterware, as well as operating system groups and users. You can use CVU for preinstallation checks and for post-installation checks of your cluster environment. CVU is especially useful during preinstallation and during installation of Oracle Clusterware and Oracle RAC components. The OUI runs CVU after installing Oracle Clusterware and Oracle Database to verify your environment.

  Install and use CVU before you install Oracle RAC to ensure that your configuration meets the minimum Oracle RAC installation requirements. Also use the CVU for ongoing administrative tasks, such as node addition and node deletion.
- DBCA—DBCA is the recommended method for creating and initially configuring Oracle RAC databases.
- NETCA—Configures the network for your Oracle RAC environment.

**See Also:**

- Chapter 3, "Administering Database Instances and Cluster Databases" for an introduction to Oracle RAC administration using Oracle Enterprise Manager, SQL*Plus, and the SRVCTL utility
- "Monitoring Oracle Real Application Clusters and Oracle Clusterware" on page 12-2
- Appendix A, "Server Control Utility Reference" for SRVCTL reference information
- *Oracle Clusterware Administration and Deployment Guide* for information about Oracle Clusterware tools such as the OIFCFG tool for allocating and deallocating network interfaces, the OCRCONFIG command-line tool for managing the Oracle Cluster Registry, and the Cluster Verification Utility (CVU)
- *Oracle Database Net Services Administrator's Guide* for more information about NETCA

## About Monitoring Oracle Real Application Clusters Environments

Web-based Enterprise Manager Database Control and Grid Control enable you to monitor an Oracle RAC database. The Enterprise Manager Console is a central point of control for the Oracle environment that you access by way of a graphical user interface (GUI). See Monitoring Oracle Real Application Clusters and Oracle Clusterware on page 12-2 and the *Oracle Database 2 Day + Real Application Clusters Guide*, *Oracle Enterprise Manager Concepts*, for detailed information about using Enterprise Manager to monitor Oracle RAC environments.



ORCL01972364

Also, note the following recommendations about monitoring Oracle RAC environments:

- Use the Enterprise Manager Console to initiate cluster database management tasks.
- Use Enterprise Manager Grid Control to administer multiple Oracle RAC databases.
- Use the global views, or `GV$` views, which are based on `V$` views. The `catclustdb.sql` script creates the `GV$` views. Run this script if you do not create your database with DBCA. Otherwise, DBCA runs this script for you.
- Use the sophisticated management and monitoring features of the Oracle Database Diagnostic and Tuning packs that include the Automatic Database Diagnostic Monitor (ADDM) and AWR.

**Note:** Although Statspack is available for backward compatibility, Statspack provides reporting only. You must run Statspack at level 7 to collect statistics related to block contention and segment block waits.

**See Also:** The *Oracle Database Performance Tuning Guide* describes the Oracle automatic features for performance diagnosing and tuning, including ADDM.

## About Evaluating Performance in Oracle Real Application Clusters Environments

You do not need to perform special tuning for Oracle RAC; Oracle RAC scales without special configuration changes. If your application performed well on a single-instance Oracle database, then it will perform well in an Oracle RAC environment. Many of the tuning tasks that you would perform on a single-instance Oracle database can also improve Oracle RAC database performance. This is especially true if your environment required scalability across a greater number of CPUs.

Some of the performance features specific to Oracle RAC include:

- Dynamic Resource Allocation
  - Oracle dynamically allocates Cache Fusion resources as needed
  - The dynamic mastering of resources improves performance by keeping resources local to data blocks
- Cache Fusion Enables A Simplified Tuning Methodology
  - You do not have to tune any parameters for Cache Fusion
  - No application-level tuning is necessary
  - You can use a bottom-up tuning approach with virtually no effect on your existing applications
- More Detailed Performance Statistics
  - More views for Oracle RAC performance monitoring
  - Enterprise Manager Database Control and Grid Control are Integrated with Oracle RAC



ORCL01972365

ORCL01972366

# 4

# Introduction to Automatic Workload Management

This chapter describes how to manage workloads in Oracle Real Application Clusters (Oracle RAC) to provide high availability and scalability for your applications. This chapter contains the following topics:

- Overview of Automatic Workload Management
- Automatic Workload Repository
- Service Deployment Options
- Fast Application Notification
- Load Balancing Advisory
- Oracle Clients That Are Integrated with Fast Application Notification
- Enabling Event Notification for Connection Failures in Oracle Real Application Clusters
- Services and Distributed Transaction Processing in Oracle Real Application Clusters
- Enabling Distributed Transaction Processing for Services
- Administering Services
- Administering Services with Enterprise Manager, PL/SQL, and SRVCTL
- Measuring Performance by Service Using the Automatic Workload Repository
- Service Thresholds and Alerts

## Overview of Automatic Workload Management

Automatic workload management enables you to manage workload distributions to provide optimal performance for users and applications. Automatic workload management comprises the following:

- **Services**—Oracle Database provides a powerful automatic workload management facility, called services, to enable the enterprise grid vision. Services are entities that you can define in Oracle RAC databases that enable you to group database workloads and route work to the optimal instances that are assigned to offer the service.
- **Connection Load Balancing**—A feature of Oracle Net Services that balances incoming connections across all of the instances that provide the requested database service.

- **High Availability Framework**—A Oracle RAC component that enables the Oracle Database to maintain components in a running state at all times.
- **Fast Application Notification (FAN)**—The notification mechanism that Oracle RAC uses to quickly alert applications about cluster state changes and workload service level changes, such as `UP` and `DOWN` events for instances, services, or nodes.
- **Load Balancing Advisory**—Provides information to applications about the current service levels that the database and its instances are providing. The load balancing advisory makes recommendations to applications about where to direct application requests to obtain the best service based on the policy that you have defined for that service.
- **Automatic Workload Repository (AWR)**—Tracks service level statistics as metrics. Server generated alerts can be placed on these metrics when they exceed or fail to meet certain thresholds. You can then respond, for example, by changing the priority of a job, stopping overloaded processes, or by modifying a service level requirement. This enables you to maintain continued service availability despite service level changes. You can configure the service level for one service to have priorities relative to other services, and you can also configure:
  - The measurement of service quality
  - Event notification and alert mechanisms to monitor service quality changes
  - Recovery scenarios for responses to service quality changes

  The Automatic Workload Repository ensures that the Oracle Clusterware workload management framework and resource manager have persistent and global representations of performance data. This information helps Oracle schedule job classes by service and to assign priorities to consumer groups. If necessary, you can rebalance workloads manually with the `DBMS_SERVICE.DISCONNECT_SESSION` PL/SQL procedure. You can also use this procedure to disconnect a series of sessions and leave the service running.

  **See Also:** *Oracle Database Performance Tuning Guide* for details about the Automatic Workload Repository and *Oracle Database PL/SQL Packages and Types Reference* for details about Oracle packages
- **Fast Connection Failover**—This is the ability of Oracle Clients to provide rapid failover of connections by subscribing to FAN events.
- **Runtime Connection Load Balancing**—This is the ability of Oracle Clients to provide intelligent allocations of connections in the connection pool based on the current service level provided by the database instances when applications request a connection to complete some work.

When a user or application connects to a database, Oracle recommends that you specify a service in the connect data portion of the connect string. Oracle Database automatically creates one database service when the database is created. For many installations, this may be all you need. To enable more flexibility in the management of the workload using the database, Oracle Database enables you to create multiple services and specify which instances offer the services. Continue reading this chapter to understand the added features that you can use with services if you are interested in greater workload management flexibility.



ORCL01972400

> **Note:** The features discussed in this chapter do not work with the default database service. You must create cluster managed services to take advantage of these features. You can only manage services that you create. Any service created by the database server will be managed by the database server.

You can deploy Oracle RAC and single-instance Oracle database environments to use automatic workload management features in many different ways. Depending on the number of nodes and your environment's complexity and objectives, your choices for optimal automatic workload management and high-availability configuration depend on several considerations that this chapter describes. The following section describes the various service deployment options.

## Automatic Workload Repository

The **Automatic Workload Repository (AWR)** tracks service level statistics as metrics. Server generated alerts can be placed on these metrics when they exceed or fail to meet certain thresholds. You can then respond, for example, by changing the priority of a job, stopping overloaded processes, or by modifying a service level requirement. This enables you to maintain continued service availability despite service level changes. You can configure the service level for one service to have priorities relative to other services, and you can also configure:

## Service Deployment Options

This section describes the following service deployment topics:

- Using Oracle Services
- Default Service Connections
- Connection Load Balancing

### Using Oracle Services

To manage workloads or a group of applications, you can define services that you assign to a particular application or to a subset of an application's operations. You can also group work by type under services. For example, online users can be a service while batch processing can be another and reporting can be yet another service type.

Oracle recommends that all users who share a service have the same service level requirements. You can define specific characteristics for services and each service can be a separate unit of work. There are many options that you can take advantage of when using services. Although you do not have to implement these options, using them helps optimize application performance.

When you define a service, you define which instances normally support that service. These are known as the `PREFERRED` instances. You can also define other instances to support a service if the service's preferred instance fails. These are known as `AVAILABLE` instances.

When you specify `PREFERRED` instances, you are specifying the number of instances on which a service will normally run. Oracle Clusterware attempts to ensure that the service always runs on the number of instances for which you have configured the service. Afterwards, due to either instance failure or planned service relocations, a service may be running on an `AVAILABLE` instance. You cannot control which



ORCL01972401

AVAILABLE instance to which Oracle Clusterware relocates the services if there is more than one in the list. When a service moves to an AVAILABLE instance, Oracle does not move the service back to the PREFERRED instance when the PREFERRED instance restarts because:

- The service is already running on the desired number of instances
- Maintaining the service on the current instance provides a higher level of service availability
- Not moving the service back to the initial PREFERRED instance prevents a second outage

You can, however, easily automate fail back by using FAN callouts.

Resource profiles are automatically created when you define a service. A resource profile describes how Oracle Clusterware should manage the service and which instance the service should failover to if the preferred instance stops. Resource profiles also define service dependencies for the instance and the database. Due to these dependencies, if you stop a database, then the instances and services are automatically stopped in the correct order.

Services are integrated with Resource Manager, which enables you to restrict the resources that are used by the users who connect with a service in an instance. The Resource Manager enables you to map a consumer group to a service so that users who connect with the service are members of the specified consumer group. Also, the Automatic Workload Repository (AWR) enables you to monitor performance by service.

Oracle Net Services provides connection load balancing to enable you to spread user connections across all of the instances that are supporting a service. For each service, you can define the method that you want the listener to use for load balancing by setting the connection load balancing goal, CLB_GOAL. You can also specify a single transparent application failover (TAF) policy for all users of a service by defining the FAILOVER_METHOD, FAILOVER_TYPE, and so on. (See the *Oracle Database Net Services Administrator's Guide* for more information about configuring TAF.)

Oracle RAC uses FAN to notify applications about configuration changes and the current service level that is provided by each instance where the service is enabled. FAN has two methods for publishing events to clients, the Oracle Notification Service (ONS) that is used by Java Database Connectivity (JDBC) clients including the Oracle Application Server, and Oracle Streams, and Advanced Queueing that is used by Oracle Call Interface (OCI) and Oracle Data Provider for .NET (ODP.NET) clients. When using Advanced Queueing, you must enable the service to use the queue by setting AQ_HA_NOTIFICATIONS to true.

With runtime connection load balancing, applications can use load balancing advisory events to provide better service to users. The Oracle JDBC and ODP.NET clients are automatically integrated to take advantage of load balancing advisory events. The load balancing advisory informs the client about the current service level that an instance is providing for a service. The load balancing advisory also recommends how much of the workload should be sent to that instance. In addition, Oracle Net Services provides connection load balancing to enable you to spread user connections across all of the instances that support a service. To enable the load balancing advisory, set the GOAL parameter on the service.

Distributed transaction processing applications have unique requirements. To make it easier to use Oracle RAC with global transactions, set the distributed transaction processing parameter on the service so that all tightly coupled branches of a distributed transaction processing transaction are run on the same instance.



ORCL01972402

**See Also:** "Services and Distributed Transaction Processing in Oracle Real Application Clusters" on page 4-22 for more information about distributed transaction processing in Oracle RAC

## Default Service Connections

A special Oracle database service is created by default for your Oracle RAC database. This default service is always available on all instances in an Oracle RAC environment, unless an instance is in restricted mode. You cannot alter this service or its properties. The database also supports the following two internal services:

- SYS$BACKGROUND is used by the background processes only
- SYS$USERS is the default service for user sessions that are not associated with any application service

Both of these internal services support all of the automatic workload management features. You cannot stop or disable either of these internal services.

**Note:** You can explicitly manage only the services that you create. If a feature of the database creates an internal service, you cannot manage it using the information in this chapter.

## Connection Load Balancing

Oracle Net Services provides the ability to balance client connections across the instances in an Oracle RAC configuration. There are two types of load balancing that you can implement: client-side and server-side load balancing. Client-side load balancing balances the connection requests across the listeners. With server-side load balancing, the listener directs a connection request to the best instance currently providing the service by using the load balancing advisory. In an Oracle RAC database, client connections should use both types of connection load balancing.

FAN, Fast Connection Failover, and the load balancing advisory depend on an accurate connection load balancing configuration that includes setting the connection load balancing goal for the service. You can use a goal of either LONG or SHORT for connection load balancing. These goals have the following characteristics:

- **LONG**—Use the LONG connection load balancing method for applications that have long-lived connections. This is typical for connection pools and SQL*Forms sessions. LONG is the default connection load balancing goal. The following is an example of modifying a service, POSTMAN, with the PL/SQL DBMS_SERVICE package and the CLB_GOAL_LONG package constant to define the connection load balancing goal for long-lived sessions:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE (service_name => 'POSTMAN'
        , clb_goal => DBMS_SERVICE.CLB_GOAL_LONG);
```

- **SHORT**—Use the SHORT connection load balancing method for applications that have short-lived connections. The following example modifies the service known as ORDER, using the DBMS_SERVICE.CLB_GOAL_SHORT package constant to set the goal to SHORT:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE (service_name => 'ORDER'
        , CLB_GOAL => DBMS_SERVICE.CLB_GOAL_SHORT);
```

When you create an Oracle RAC database with the DBCA, it automatically:

- Configures and enables server-side load balancing



ORCL01972403

- Sets the local and remote listener parameters (**Note:** If you do not use DBCA, you should set the `LOCAL_LISTENER` and `REMOTE_LISTENER` database parameters manually, especially if you do not use port 1521.)
- Creates a sample client-side load balancing connection definition in the `tnsnames.ora` file on the server

When Oracle Net Services establishes a connection to an instance, the connection remains open until the client closes the connection, the instance is shutdown, or a failure occurs. If you configure TAF for the connection, then Oracle Database moves the session to a surviving instance when an outage occurs.

TAF can restart a query after failover has completed but for other types of transactions, such as `INSERT`, `UPDATE`, or `DELETE`, the application must rollback the failed transaction and resubmit the transaction. You must reexecute any session customizations, in other words, `ALTER SESSION` statements, after failover has occurred. However, with TAF, a connection is not moved during normal processing, even if the workload changes over time.

Services simplify the deployment of TAF. You can define a TAF policy for a service and all connections using this service will automatically have TAF enabled. This does not require any client-side changes. The TAF setting on a service overrides any TAF setting in the client connection definition. To define a TAF policy for a service, use the `DBMS_SERVICE` PL/SQL procedure as in the following example:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE (service_name => 'gl.us.oracle.com'
, aq_ha_notifications => TRUE
, failover_method => DBMS_SERVICE.FAILOVER_METHOD_BASIC
, failover_type => DBMS_SERVICE.FAILOVER_TYPE_SELECT
, failover_retries => 180
, failover_delay => 5
, clb_goal => DBMS_SERVICE.CLB_GOAL_LONG);
```

> **See Also:** *Oracle Database Net Services Administrator's Guide* for more information about configuring TAF

Client-side load balancing is defined in your client connection definition by setting the parameter `LOAD_BALANCE=ON` (the default is `ON` for description lists). When you set this parameter to `ON`, Oracle randomly selects an address in the address list, and connects to that node's listener. This balances client connections across the available listeners in the cluster. When the listener receives the connection request, the listener connects the user to an instance that the listener knows provides the requested service. To see what services a listener supports, run the `lsnrctl services` command.

In addition to client-side load balancing, Oracle Net Services include connection failover. If an error is returned from the chosen address in the list, Oracle Net Services will try the next address in the list until it is either successful or it has exhausted all addresses in its list. To increase availability, you can specify a timeout within which Oracle Net will wait for a response from the listener.

You can avoid delays by setting the `oracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STR` property, as follows:

```
Properties prop = new Properties ();
prop.put (oracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STR,
"" + (1 * 1000)); // 1 second
dbPools[ poolIndex ].setConnectionProperties ( prop );
```



ORCL01972404

The parameter value is specified in milliseconds. Therefore, it is possible to reduce the timeout to 500Ms if the application retries connecting.

For OCI clients, create a local `sqlnet.ora` file on the client side. Configure the connection timeout in this file by adding the following line:

```
sqlnet.outbound_connect_timeout = 1
```

The granularity of the timeout value for the OCI client is in seconds.

---

**Note:** Do *not* configure the connection timeout in the `sqlnet.ora` file on the server.

---

**See Also:** *Oracle Database Net Services Administrator's Guide* for detailed information about both types of load balancing

# Fast Application Notification

This section provides a detailed description of FAN under the following topics:

- Overview of Fast Application Notification
- Application High Availability with Services and FAN
- Managing Unplanned Outages
- Managing Planned Outages
- Fast Application Notification High Availability Events
- Using Fast Application Notification Callouts

**See Also:** "Oracle Clients That Are Integrated with Fast Application Notification" on page 4-12 for more information about specific client environments that you can use with FAN

## Overview of Fast Application Notification

FAN is a notification mechanism that Oracle RAC uses to notify other processes about configuration and service level information that includes service status changes, such as UP or DOWN events. Applications can respond to FAN events and take immediate action. FAN `UP` and `DOWN` events can apply to instances, services, and nodes.

For cluster configuration changes, the Oracle RAC high availability framework publishes a FAN event immediately when a state change occurs in the cluster. Instead of waiting for the application to poll the database and detect a problem, applications can receive FAN events and react immediately.

FAN also publishes load balancing advisory events. Applications can take advantage of the load balancing advisory FAN events to direct work requests to the instance in the cluster that is currently providing the best service quality. You can take advantage of FAN events in the following three ways:

1. Your application can use FAN without programmatic changes if you use an integrated Oracle client. The integrated clients for FAN events include Oracle Database JDBC, Oracle Database ODP.NET, and Oracle Database OCI. This includes applications that use TAF. The integrated Oracle clients must be Oracle Database 10*g* Release 2 or later to take advantage of the load balancing advisory FAN events. (See the *Oracle Database Net Services Administrator's Guide* for more information about configuring TAF.)



ORCL01972405

2. Applications can use FAN programmatically by using the ONS Application Programming Interface (API) to subscribe to FAN events and to execute event handling actions upon the receipt of an event.
3. You can implement FAN with server-side callouts on your database tier.

For DOWN events, the disruption to the application can be minimized because sessions to the failed instance or node can be terminated. Incomplete transactions can be terminated and the application user is immediately notified. Application users who request connections are directed to available instances only. For UP events, when services and instances are started, new connections can be created so that the application can immediately take advantage of the extra resources. Through server-side callouts, you can also use FAN to:

- Log status information
- Page DBAs or to open support tickets when resources fail to start
- Automatically start dependent external applications that need to be co-located with a service
- Change resource plans or to shut down services when the number of available instances decreases, for example, if nodes fail
- Automate the fail back of a service to PREFERRED instances if needed

FAN events are published using ONS and an Oracle Streams Advanced Queuing. The publication mechanisms are automatically configured as part of your Oracle RAC installation.

The Connection Manager (CMAN) and Oracle Net Services listeners are integrated with FAN events. This enables the listener and CMAN to immediately de-register services provided by the failed instance and to avoid erroneously sending connection requests to failed instances.

If you use the service goal CLB_GOAL_SHORT, then the listener uses the load balancing advisory when the listener balances the connection loads. When load balancing advisory is enabled, the metrics used for the listener are finer grained.

## Application High Availability with Services and FAN

Oracle Database focuses on maintaining service availability. In Oracle RAC, Oracle services are designed to be continuously available with loads shared across one or more instances. The Oracle RAC high availability framework maintains service availability by using Oracle Clusterware and resource profiles.

The Oracle RAC high availability framework monitors the database and its services and sends event notifications using FAN. Oracle Clusterware recovers and balances services according to business rules.

## Managing Unplanned Outages

You can assign services in Oracle RAC to one or more instances. If Oracle RAC detects an outage, then Oracle Clusterware isolates the failed component and recovers the dependent components. For services, if the failed component is an instance, then Oracle Clusterware relocates the service to an available instance in the cluster. FAN events can occur at various levels within the Oracle Database architecture and are published through ONS and Advanced Queuing. You can also program notification using FAN callouts.



ORCL01972406

> **Note:** Oracle Database does not run Oracle RAC callouts with guaranteed ordering. Callouts are run asynchronously and they are subject to scheduling variabilities.

Notification occurs from a surviving node when the failed node is out of service. The location and number of instances in an Oracle RAC environment that provide a service are transparent to applications. Restart and recovery are automatic, including the restarting of the subsystems, such as the listener and the Automatic Storage Management (ASM) processes, not just the database. You can use FAN callouts to report faults to your fault management system and to initiate repair jobs.

## Managing Planned Outages

For repairs, upgrades, and changes that require you to isolate one or more instances, Oracle RAC provides interfaces that relocate, disable, and enable services to minimize service disruption to application users. Once you complete the operation, you can return the service to normal operation.

Due to dependencies, if you manually shutdown your database, then all of your services automatically stop. If you then manually restart the database, then you must also restart the database's services. Use FAN callouts to automate starting the services when the database starts.

## Fast Application Notification High Availability Events

Table 4–1 describes the FAN event record parameters and the event types, followed by name-value pairs for the event properties. The event type is always the first entry and the timestamp is always the last entry as in the following example:

```
FAN event type: service_member
Properties: version=1.0 service=ERP database=FINPROD instance=FINPROD3 host=node3
status=up
```

***Table 4–1 Event Record Parameters and Descriptions***

| Parameter | Description |
|---|---|
| VERSION | Version of the event record. Used to identify release changes. |
| EVENT TYPE | SERVICE, SERVICE_MEMBER, DATABASE, INSTANCE, NODE, ASM, SRV_PRECONNECT. Note that Database and Instance types provide the database service, such as DB_UNIQUE_NAME.DB_DOMAIN. |
| DATABASE UNIQUE NAME | The unique database supporting the service; matches the initialization parameter value for DB_UNIQUE_NAME, which defaults to the value of the initialization parameter DB_NAME. |
| INSTANCE | The name of the instance that supports the service; matches the ORACLE_SID value. |
| NODE NAME | The name of the node that supports the service or the node that has stopped; matches the node name known to Cluster Synchronization Services (CSS). |
| SERVICE | The service name; matches the service in DBA_SERVICES. |
| STATUS | Values are UP, DOWN, NOT_RESTARTING, PRECONN_UP, PRECONN_DOWN, and UNKNOWN. |
| REASON | Failure, Dependency, User, Autostart, Restart. |
| CARDINALITY | The number of service members that are currently active; included in all UP events. |
| INCARNATION | For node DOWN events; the new cluster incarnation. |



ORCL01972407

*Table 4–1 (Cont.) Event Record Parameters and Descriptions*

| Parameter | Description |
|---|---|
| TIMESTAMP | The local time zone to use when ordering notification events. |

A FAN record matches the database signature of each session as shown in Table 4–2.

*Table 4–2 FAN Parameters and Matching Database Signatures*

| FAN Parameter | Matching Oracle Database Signature |
|---|---|
| SERVICE | sys_context('userenv', 'service_name') |
| DATABASE UNIQUE NAME | sys_context('userenv', 'db_unique_name') |
| INSTANCE | sys_context('userenv', 'instance_name') |
| CLUSTER NODE NAME | sys_context('userenv', 'server_host') |

## Using Fast Application Notification Callouts

FAN callouts are server-side executables that Oracle RAC executes immediately when high availability events occur. You can use FAN callouts to automate the following activities when events occur in a cluster configuration, such as:

- Opening fault tracking tickets
- Sending messages to pagers
- Sending e-mail
- Starting and stopping server-side applications
- Maintaining an uptime log by logging each event as it occurs
- Relocating low-priority services when high priority services come online

To use FAN callouts, place an executable in the directory *CRS_home*`/racg/usrco` on every node that runs Oracle Clusterware. If you are using scripts, then set the shell as the first line of the executable. The following is an example file for the *CRS_home*`/racg/usrco/callout.sh` callout:

```
#! /bin/ksh
FAN_LOGFILE= [your path name]/admin/log/`hostname`_uptime.log
echo $* "reported="`date` >> $FAN_LOGFILE &
```

The following output is from the previous example:

```
NODE VERSION=1.0 host=sun880-2 incarn=23 status=nodedown reason=
timestamp=08-Oct-2004 04:02:14 reported=Fri Oct 8 04:02:14 PDT 2004
```

Example callout scripts are available in the Oracle Real Application Clusters Sample Code section on Oracle Technology Network at `http://www.oracle.com/technology/sample_code/products/rac/`

> **See Also:** Table 4–1, " Event Record Parameters and Descriptions" for information about the callout and event details

A FAN record matches the database signature of each session, as shown in Table 4–2, " FAN Parameters and Matching Database Signatures". The signature information is also available using `OCI_ATTRIBUTES`. These attributes are available in the OCI



ORCL01972408

Connection Handle. Use this information to take actions on sessions that match the FAN event data.

# Load Balancing Advisory

This section describes the load balancing advisory under the following topics:

- Overview of the Load Balancing Advisory
- Configuring Your Environment to Use the Load Balancing Advisory
- Load Balancing Advisory FAN Events

## Overview of the Load Balancing Advisory

Load balancing distributes work across all of the available Oracle RAC database instances. Oracle recommends that applications use persistent connections that span the instances that offer a particular service. Connections are created infrequently and exist for a long duration. Work comes into the system with high frequency, borrows these connections, and exists for a relatively short duration. The load balancing advisory provides advice about how to direct incoming work to the instances that provide the optimal quality of service for that work. This minimizes the need to relocate the work later.

By using the `THROUGHPUT` or `SERVICE_TIME` goals, feedback is built in to the system. Work is routed to provide the best service times globally, and routing responds gracefully to changing system conditions. In a steady state, the system approaches equilibrium with improved throughput across all of the Oracle RAC instances.

Standard architectures that can use the load balancing advisory include connection load balancing, transaction processing monitors, application servers, connection concentrators, hardware and software load balancers, job schedulers, batch schedulers, and message queuing systems. All of these applications can allocate work.

The load balancing advisory is deployed with key Oracle clients, such as a listener, the JDBC Implicit Connection Cache, and the ODP.NET Connection Pool. The load balancing advisory is also open for third-party subscription by way of ONS.

## Configuring Your Environment to Use the Load Balancing Advisory

You can configure your environment to use the load balancing advisory by defining service-level goals for each service for which you want to enable load balancing. This enables the load balancing advisory for that service and FAN load balancing events are published. There are two types of service-level goals for runtime:

- `SERVICE TIME`—Attempts to direct work requests to instances according to response time. Load balancing advisory data is based on elapsed time for work done in the service plus available bandwidth to the service. An example for the use of `SERVICE TIME` is for workloads such as internet shopping where the rate of demand changes:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE (service_name => 'OE' -
, goal => DBMS_SERVICE.GOAL_SERVICE_TIME -
, clb_goal => DBMS_SERVICE.CLB_GOAL_SHORT);
```

- `THROUGHPUT`—Attempts to direct work requests according to throughput. The load balancing advisory is based on the rate that work is completed in the service plus available bandwidth to the service. An example for the use of `THROUGHPUT` is



ORCL01972409

for workloads such as batch processes, where the next job starts when the last job completes:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE (service_name => 'sjob' -
        , goal => DBMS_SERVICE.GOAL_THROUGHPUT -
       , clb_goal => DBMS_SERVICE.CLB_GOAL_LONG);
```

Setting the goal to NONE disables load balancing for the service. You can see the goal settings for a service in the data dictionary and in the DBA_SERVICES, V$SERVICES, and V$ACTIVE_SERVICES views.

**See Also:** "Administering Services" on page 4-24 for more information about administering services and adding goals to services

## Load Balancing Advisory FAN Events

The load balancing advisory FAN events provide metrics for load balancing algorithms. The easiest way to take advantage of these events is to use the runtime connection load balancing feature of an Oracle integrated client such as JDBC, ODP.NET, or OCI. Client applications can subscribe to these events directly by way of the ONS API. Table 4–3 describes the load balancing advisory FAN event parameters.

***Table 4–3 Load Balancing Advisory FAN Events***

| Parameter | Description |
|---|---|
| VERSION | Version of the event record. Used to identify release changes. |
| EVENT TYPE | SERVICE, SERVICE_MEMBER, DATABASE, INSTANCE, NODE, ASM, SRV_PRECONNECT. Note that Database and Instance types provide the database service, such as DB_UNIQUE_NAME.DB_DOMAIN. |
| SERVICE | The service name; matches the service in DBA_SERVICES. |
| DATABASE UNIQUE NAME | The unique database supporting the service; matches the initialization parameter value for DB_UNIQUE_NAME, which defaults to the value of the initialization parameter DB_NAME. |
| INSTANCE | The name of the instance that supports the service; matches the ORACLE_SID value. |
| PERCENT | The percentage of work requests to send to this database instance. |
| FLAG | Indication of the service quality relative to the service goal. Valid values are GOOD, VIOLATING, NO DATA, and BLOCKED. |
| TIMESTAMP | The local time zone to use when ordering notification events. |

Use the following example to monitor load balancing advisory events:

```
SET PAGES 60 COLSEP '|' LINES 132 NUM 8 VERIFY OFF FEEDBACK OFF
COLUMN user_data HEADING "AQ Service Metrics" FORMAT A60 WRAP
BREAK ON service_name SKIP 1
SELECT
 TO_CHAR(enq_time, 'HH:MI:SS') Enq_time, user_data
 FROM sys.sys$service_metrics_tab
 ORDER BY 1 ;
```

# Oracle Clients That Are Integrated with Fast Application Notification

Oracle has integrated FAN with many of the common client application environments that are used to connect to Oracle RAC databases. Therefore, the easiest way to use FAN is to use an integrated Oracle Client.



ORCL01972410

You can use the OCI Session Pools, CMAN session pools, and JDBC and ODP.NET connection pools. OCI applications with TAF enabled should use FAN high availability events for fast failover. The overall goal is to enable applications to consistently obtain connections to available instances that provide the best service. Due to the integration with FAN, Oracle integrated clients are more aware of the current status of an Oracle RAC cluster. This prevents client connections from waiting or trying to connect to an instance that is no longer available.

When instances start, Oracle RAC uses FAN to notify the connection pool so that the connection pool can create connections to the recently started instance and take advantage of the additional resources that this instance provides. The use of connection pools and FAN requires that you have properly configured database connection load balancing across all of the instances that provide the service(s) that the connection pool is using. Oracle recommends that you configure both client-side and server-side load balancing with Oracle Net Services, which is the default when you use DBCA to create your database. Oracle connection pools that are integrated with FAN can:

- Balance connections across all of the Oracle RAC instances when a service starts; this is preferable to directing the sessions that are defined for the connection pool to the first Oracle RAC instance that supports the service
- Remove terminated connections immediately when a service is declared `DOWN` at an instance, and immediately when nodes are declared `DOWN`
- Report errors to clients immediately when Oracle detects the `NOT RESTARTING` state, instead of making the client wait while the service repeatedly attempts to restart
- Balance work requests at run time using load balancing advisory events

The next sections describe how to enable FAN events for the several specific client development environments:

- Enabling JDBC Clients for Fast Connection Failover
- Enabling Oracle Call Interface Clients for Fast Connection Failover
- Enabling Oracle Call Interface Clients for Runtime Connection Load Balancing
- Enabling ODP.NET Clients to Receive FAN High Availability Events
- Enabling ODP.NET Clients to Receive FAN Load Balancing Advisory Events

## Enabling JDBC Clients for Fast Connection Failover

Enabling FAN for the Oracle JDBC Implicit Connection Cache enables FAN high availability events and the load balancing advisory. Your application can use the JDBC development environment for either thick or thin JDBC clients to use FAN. You must use the JDBC Implicit Connection Cache to enable the FAN features of Fast Connection Failover and runtime connection load balancing.

To configure the JDBC client, set the `FastConnectionFailoverEnabled` property before making the first `getConnection()` request to a data source. When you enable Fast Connection Failover, the failover applies to every connection in the connection cache. If your application explicitly creates a connection cache using the Connection Cache Manager, then you must first set `FastConnectionFailoverEnabled`.

> **See Also:** The *Oracle Database JDBC Developer's Guide and Reference* chapter about "Implicit Connection Caching" for more information about configuring the JDBC implicit connection cache and ONS



ORCL01972411

### Using FAN with Thick JDBC and Thin JDBC Clients

This procedure explains how to enable FAN events for JDBC. For thick JDBC clients, if you enable Fast Connection Failover, do not enable TAF, either on the client or for the service. Enabling Fast Connection Failover with thin or thick JDBC clients enables the connection pool to receive and react to all FAN events. To enable Fast Connection Failover, you must first enable the Implicit Connection Cache as described in the following procedure:

1. On a cache enabled DataSource, set the DataSource property `FastConnectionFailoverEnabled` to `true` as in the following example to enable FAN for Oracle implicit JDBC connect cache:

```
OracleDataSource ods = new OracleDataSource()
...
ods.setConnectionCachingEnabled(True);
ods.setFastConnectionFailoverEnabled(True);
ods.setConnectionCacheName("MyCache");
ods.setConnectionCacheProperties(cp);
ods.setURL("jdbc:oracle:thin:@(DESCRIPTION=
   (LOAD_BALANCE=on)
   (ADDRESS=(PROTOCOL=TCP)(HOST=VIP1)(PORT=1521))
   (ADDRESS=(PROTOCOL=TCP)(HOST=VIP2)(PORT=1521))
   (CONNECT_DATA=(service_name=service_name)))");
```

> **Note:** Use the following system property to enable FAN without making data source changes: `-D oracle.jdbc.FastConnectionFailover=true`.

2. Ensure that you have configured ONS on each node that is running Oracle Clusterware as in the following example using the *CRS_home*`/opmn/conf/ons.config` file.

> **Note:** This should have been automatically completed during the Oracle RAC installation.

```
localport=6100 # This is the port ONS is writing to on this node
remoteport=6200 # This is the port ONS is listening on this node
loglevel=3
useocr=on
```

The information in the Oracle Cluster Registry (OCR) for ONS daemons is automatically configured by DBCA. If you are upgrading from a previous version of the database, then manually configure the OCR as follows. To add the middle tier nodes or to update the Oracle RAC nodes, use `racgons` from the Oracle Clusterware `bin` directory:

- To add the ONS daemon configuration:

```
racgons add_config hostname:port [hostname:port] ...
```

- To remove the ONS daemon configuration:

```
racgons remove_config hostname[:port] [hostname:port]..
```

3. Configure remote ONS subscription. Remote ONS subscription offers the following advantages:



ORCL01972412

- Support for an All Java mid-tier software
- An ONS daemon is not needed on the client system you do not need to manage this process
- Simple configuration by way of a `DataSource` property

When using remote ONS subscription for Fast Connection Failover, an application uses `setONSConfiguration`, using the string `remoteONSConfig`, on an Oracle DataSource instance as in the following example:

```
ods.setONSConfiguration("nodes=racnode1:4200,racnode2:4200");
```

4. When you start the application, make sure that the `ons.jar` file is located on the application `CLASSPATH`. The `ons.jar` file is part of the Oracle client installation.

**See Also:** *Oracle Database JDBC Developer's Guide and Reference* for more information about JDBC

## Enabling Oracle Call Interface Clients for Fast Connection Failover

OCI clients can enable Fast Connection Failover by registering to receive notifications about Oracle RAC high availability FAN events and respond when events occur. This improves the session failover response time in OCI and also removes terminated connections from connection and session pools. This feature works on OCI applications, including those that use TAF, connection pools, or session pools.

First, you must enable a service for high availability events. This automatically populates the advanced queuing `ALERT_QUEUE`. If your application is using TAF, then enable the TAF settings for the service. Configure client applications to connect to an Oracle RAC database. Clients can register callbacks that are used whenever an event occurs. This reduces the time that it takes to detect a connection failure. During `DOWN` event processing, OCI:

- Terminates affected connections at the client and returns an error
- Removes connections from the OCI connection pool and the OCI session pool—the session pool maps each session to a physical connection in the connection pool, and there can be multiple sessions for each connection
- Fails over the connection if you have configured TAF

If TAF is not configured, then the client only receives an error.

**Note:** OCI does not manage `UP` events.

Perform the following steps to configure Fast Connection Failover with an OCI client:

1. Ensure that the service that you are using has Advanced Queuing notifications enabled by setting the services' values of `AQ_HA_NOTIFICATIONS` to `TRUE`. For example:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE (service_name => 'gl.us.oracle.com'
        , aq_ha_notifications => TRUE
        , failover_method  => DBMS_SERVICE.FAILOVER_METHOD_BASIC
        , failover_type    => DBMS_SERVICE.FAILOVER_TYPE_SELECT
        , failover_retries => 180
        , failover_delay   => 5
        , clb_goal => DBMS_SERVICE.CLB_GOAL_LONG);
```

2. Enable `OCI_EVENTS` at environment creation time on the client as follows:



ORCL01972413

```
( OCIEnvCreate(...) )
```

3. Client applications must link with the client thread or operating system library.
4. Optionally register a client `EVENT` callback

To see the alert information, you can query the views `DBA_OUTSTANDING_ALERTS` and `DBA_ALERT_HISTORY`.

> **See Also:** *Oracle Call Interface Programmer's Guide* for more information about OCI and the *Oracle Database Net Services Administrator's Guide* for more information about configuring TAF

## Enabling Oracle Call Interface Clients for Runtime Connection Load Balancing

As of Oracle Database 11*g* Release 1 (11.1), Oracle Call Interface (OCI) session pooling enables multiple threads of an application to use a dynamically managed set of pre-created database sessions. Oracle continually reuses the sessions in the pool to form nearly permanent channels to the instances, thus saving the overhead of creating and closing sessions every time applications need them.

To fully optimize session pools with Oracle RAC, sessions given to the requesting threads are appropriately mapped to instances based on the instance load. The OCI session pools use service metrics information received from the Load Balancing Advisory in Oracle RAC to allocate the sessions, resulting in better performance

OCI dynamically maps sessions to the threads based on the instance load. It uses service metrics provided by the Load Balancing Advisory to perform runtime connection load balancing.[1] The number of connections can exceed the number of threads at certain times, but the OCI eventually adjusts this automatically, without any need for user intervention.

If the session pool supports services that span multiple instances, then the work requests are distributed across instances so that the instances providing better service are given more requests. For session pools that support services at only one instance, the first available session in the pool is adequate.

Runtime connection load balancing is enabled by default in a Oracle Database Release 11.1 (or higher) client talking to a server running Oracle Database Release 10.2 (or higher). To disable runtime connection load balancing, set the mode parameter to `OCI_SPC_NO_RLB` when calling `OCISessionPoolCreate()`.

To receive the service metrics based on the service time, ensure you have met the following conditions:

1. Link the application with the threads library.
2. Create the OCI environment in `OCI_EVENTS` and `OCI_THREADED` mode.
3. Enable event notification for the service for which the session pool is created.

   To enable event notification, use the `DBMS_SERVICE.MODIFY_SERVICE` procedure and set the Advanced Queuing `AQ_HA_NOTIFICATIONS` parameter to `TRUE`.
4. Modify the service to set up its service goal and `clb_goal`, as shown in the following example for the `myService` service:

```
EXEC DBMS_SERVICE.MODIFY_SERVICE(''myService'',
```

[1] Runtime connection load balancing is basically routing work requests to sessions in a session pool that can best serve the work. It comes into effect when selecting a session from an existing session pool. Thus, runtime connection load balancing is a very frequent activity.



ORCL01972414

```
DBMS_SERVICE.GOAL_SERVICE_TIME,
DBMS_SERVICE.CLB_GOAL=SHORT);
```

## Enabling ODP.NET Clients to Receive FAN High Availability Events

Oracle Data Provider for .NET (ODP.NET) connection pools can subscribe to notifications that indicate when nodes, services, and service members are down. After a `DOWN` event, Oracle Database cleans up sessions in the connection pool that go to the instance that stops and ODP.NET proactively disposes connections that are no longer valid. ODP.NET establishes connections to existing Oracle RAC instances if the removal of severed connections brings the total number of connections below the value that is set for the `MIN_POOL_SIZE` parameter.

The procedures for enabling ODP.NET are similar to the procedures for enabling JDBC in that you must set parameters in the connection string to enable Fast Connection Failover. Perform the following steps to enable FAN:

1. Enable Advanced Queuing notifications by using the `DBMS.SERVICE` package as in the following example:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE (service_name => 'gl.us.oracle.com'
        , aq_ha_notifications => TRUE
        , clb_goal => DBMS_SERVICE.CLB_GOAL_LONG);
```

2. Execute the following for the users that will be connecting by way of the .Net Application, where *`user_name`* is the user name:

```
execute dbms_aqadm.grant_queue_privilege('DEQUEUE','SYS.SYS$SERVICE_METRICS',
user_name);
```

3. Enable Fast Connection Failover for ODP.NET connection pools by subscribing to FAN high availability events. Do this by setting the `ha events` connection string attribute to `true` at connection time. Note that this only works if you are using connection pools. In other words, do this if you have set the `pooling` attribute to `true`, which is the default. The following example shows this in more detail where *`user_name`* is the name of the user and *`password`* is the password:

```
// C#
using System;
using Oracle.DataAccess.Client;

class HAEventEnablingSample
{
  static void Main()
  {
    OracleConnection con = new OracleConnection();

    // Open a connection using ConnectionString attributes
    // Also, enable "load balancing"
    con.ConnectionString =
      "User Id=user_name;Password=password;Data Source=oracle;" +
      "Min Pool Size=10;Connection Lifetime=120;Connection Timeout=60;" +
      "HA Events=true;Incr Pool Size=5;Decr Pool Size=2";

    con.Open();

    // Create more connections and carry out work against the DB here.

    // Dispose OracleConnection object
    con.Dispose();
```



ORCL01972415

```
    }
}
```

## Enabling ODP.NET Clients to Receive FAN Load Balancing Advisory Events

Use the following procedures to enable ODP.NET to receive FAN load balancing advisory events:

1. Enable Advanced Queuing notifications by using the `DBMS.SERVICE` package, and by setting the `GOAL` and `CLB_GOAL` for the service as in the following example:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE (service_name => 'gl.us.oracle.com'
        , aq_ha_notifications => TRUE
        , failover_method  => DBMS_SERVICE.FAILOVER_METHOD_BASIC
        , failover_type    => DBMS_SERVICE.FAILOVER_TYPE_SELECT
        , failover_retries => 180
        , failover_delay   => 5
        , clb_goal => DBMS_SERVICE.CLB_GOAL_LONG);
```

2. Ensure Oracle Net Services is configured for connection load balancing.

3. Execute the following for the user that will be connecting by way of the .Net Application where *user_name* is the name of the user:

```
execute dbms_aqadm.grant_queue_privilege('DEQUEUE','SYS.SYS$SERVICE_METRICS',
user_name);
```

4. To take advantage of load balancing events with ODP.NET connection pools, set the load balancing string to `TRUE`; the default is `FALSE`. You can do this at connect time. This only works if you are using connection pools, or when the pooling attribute is set to `TRUE` which is the default, as in the following example where *user_name* is the name of the user and *password* is the password:

```
// C#
using System;
using Oracle.DataAccess.Client;

class LoadBalancingEnablingSample
{
  static void Main()
  {
    OracleConnection con = new OracleConnection();

    // Open a connection using ConnectionString attributes
    // Also, enable "load balancing"
    con.ConnectionString =
      "User Id=user_name;Password=password;Data Source=oracle;" +
      "Min Pool Size=10;Connection Lifetime=120;Connection Timeout=60;" +
      "Load Balancing=true;Incr Pool Size=5;Decr Pool Size=2";

    con.Open();

    // Create more connections and carry out work against the DB here.

    // Dispose OracleConnection object
    con.Dispose();
  }
}
```



ORCL01972416

**See Also:** *Oracle Data Provider for .NET Developer's Guide* for more information about ODP. NET and the *Oracle Database PL/SQL Packages and Types Reference* for more information about the `DBMS_SERVICES` PL/SQL package

**Note:** ODP.NET does not support connection re-distribution when a node starts. However, if you have enabled failover on the server-side, then ODP.NET can migrate connections to available instances.

## Enabling Event Notification for Connection Failures in Oracle Real Application Clusters

Event notification is enabled if the `SQL_ORCLATTR_FAILOVER_CALLBACK` and `SQL_ORCLATTR_FAILOVER_HANDLE` attributes of the `SQLSetConnectAttr` function are set when a connection failure occurs in an Oracle RAC database environment. The symbols for the new attributes are defined in the `sqora.h` file. The `SQL_ORCLATTR_FAILOVER_CALLBACK` attribute is used to specify the address of a routine to call when a failure event takes place.

The `SQL_ORCLATTR_FAILOVER_HANDLE` attribute is used to specify a context handle that will be passed as one of the parameters in the callback routine. This attribute is necessary for the ODBC application to determine which connection the failure event is taking place on.

The function prototype for the callback routine is as follows:

```
void failover_callback(void *handle, SQLINTEGER fo_code)
```

The `handle` parameter is the value that was set by the `SQL_ORCLATTR_FAILOVER_HANDLE` attribute. Null is returned if the attribute has not been set.

The `fo_code` parameter identifies the failure event that is taking place. The failure events map directly to the events defined in the OCI programming interface. The list of possible events is as follows:

- `ODBC_FO_BEGIN`
- `ODBC_FO_ERROR`
- `ODBC_FO_ABORT`
- `ODBC_FO_REAUTH`
- `ODBC_FO_END`

The following is a sample program that demonstrates how to use this feature where *`user_name`* is the name of the user and *`password`* is the password:

```
/*
NAME
ODBCCallbackTest
DESCRIPTION
Program to demonstrate the connection failover callback feature.
PUBLIC FUNCTION(S)
main
PRIVATE FUNCTION(S)
NOTES
Command Line: ODBCCallbackTest filename [odbc-driver]
*/
#include <malloc.h>
```

```
#include <stdio.h>
#include <string.h>
#include <sql.h>
#include <sqlext.h>
#include <sqora.h>
#define TRUE  1
#define FALSE 0
/*
 * Funtion Prototypes
 */
void display_errors(SQLSMALLINT HandleType, SQLHANDLE Handle);
void failover_callback(void *Handle, SQLINTEGER fo_code);
/*
 * Macros
 */
#define ODBC_STS_CHECK(sts) \
if (sts != SQL_SUCCESS) \
{ \
display_errors(SQL_HANDLE_ENV, hEnv); \
display_errors(SQL_HANDLE_DBC, hDbc); \
display_errors(SQL_HANDLE_STMT, hStmt); \
return FALSE; \
}
/*
 * ODBC Handles
 */
SQLHENV *hEnv = NULL; // ODBC Environment Handle
SQLHANDLE *hDbc = NULL; // ODBC Connection Handle
SQLHANDLE *hStmt = NULL; // ODBC Statement Handle
/*
 * MAIN Routine
 */
main(int argc, char **argv)
{
SQLRETURN rc;
/*
 * Connection Information
 */
SQLTCHAR *dsn = "odbctest";
SQLTCHAR *uid = "user_name";
SQLTCHAR *pwd = "password";
SQLTCHAR *szSelect = "select * from emp";
/*
 * Allocate handles
 */
rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, (SQLHANDLE *)&hEnv);
ODBC_STS_CHECK(rc)
rc = SQLSetEnvAttr(hEnv, SQL_ATTR_ODBC_VERSION, (SQLPOINTER)SQL_OV_ODBC3, 0);
ODBC_STS_CHECK(rc);
rc = SQLAllocHandle(SQL_HANDLE_DBC, hEnv, (SQLHANDLE *)&hDbc);
ODBC_STS_CHECK(rc);
/*
 * Connect to the database
 */
rc = SQLConnect(hDbc, dsn, (SQLSMALLINT)strlen(dsn),
uid, (SQLSMALLINT)strlen(uid),
pwd, (SQLSMALLINT)strlen(pwd));
ODBC_STS_CHECK(rc);
/*
 * Set the connection failover attributes
```



ORCL01972418

```
 */
rc = SQLSetConnectAttr(hDbc, SQL_ORCLATTR_FAILOVER_CALLBACK, &failover_
callback,0);
ODBC_STS_CHECK(rc);
rc = SQLSetConnectAttr(hDbc, SQL_ORCLATTR_FAILOVER_HANDLE, hDbc, 0);
ODBC_STS_CHECK(rc);
/*
 * Allocate the statement handle
 */
rc = SQLAllocHandle(SQL_HANDLE_STMT, hDbc, (SQLHANDLE *)&hStmt);
ODBC_STS_CHECK(rc);
/*
 * Wait for connection failovers
 */
while (TRUE)
{
  sleep(5000);
  rc = SQLExecDirect(hStmt,szSelect, strlen(szSelect));
  ODBC_STS_CHECK(rc);
  rc = SQLFreeStmt(hStmt, SQL_CLOSE);
  ODBC_STS_CHECK(rc);
}
/*
 * Free up the handles and close the connection
 */
rc = SQLFreeHandle(SQL_HANDLE_STMT, hStmt);
ODBC_STS_CHECK(rc);
rc = SQLDisconnect(hDbc);
ODBC_STS_CHECK(rc);
rc = SQLFreeHandle(SQL_HANDLE_DBC, hDbc);
ODBC_STS_CHECK(rc);
rc = SQLFreeHandle(SQL_HANDLE_ENV, hEnv);
ODBC_STS_CHECK(rc);
return TRUE;
}
/*
 * Failover Callback Routine
 */
void failover_callback(void *Handle, SQLINTEGER fo_code)
{
switch (fo_code)
{
case ODBC_FO_BEGIN:
    printf("ODBC_FO_BEGIN recevied");
        break;
case ODBC_FO_ERROR:
    printf("ODBC_FO_ERROR recevied");
        break;
case ODBC_FO_ABORT:
        printf("ODBC_FO_ABORT recevied");
        break;
case ODBC_FO_REAUTH:
        printf("ODBC_FO_REAUTH recevied");
        break;
case ODBC_FO_END:
        printf("ODBC_FO_END recevied");
        break;
default:
        printf("Invalid or unknown ODBC failover code recevied");
        break;
```



ORCL01972419

```
};
return;
}
/*
 * Retrieve the errors associated with the handle passed
 * and display them.
 */
void display_errors(SQLSMALLINT HandleType, SQLHANDLE Handle)
{
SQLTCHAR MessageText[256];
SQLTCHAR SqlState[5+1];
SQLSMALLINT i=1;
SQLINTEGER NativeError;
SQLSMALLINT TextLength;
SQLRETURN sts = SQL_SUCCESS;
if (Handle == NULL) return;
/*
 * Make sure all SQLState text is null terminated
 */
SqlState[5] = '\0';
/*
 * Fetch and display all diagnostic records that exist for this handle
 */
while (sts == SQL_SUCCESS)
{
NativeError = 0;
TextLength = 0;
sts = SQLGetDiagRec(HandleType, Handle, i, SqlState, &NativeError,
(SQLTCHAR *)&MessageText, sizeof(MessageText),&TextLength);
if (sts == SQL_SUCCESS)
{
  printf("[%s]%s\n",NativeError, &MessageText);
  if (NativeError != 0)
  {
    printf("Native Error Code: %d", NativeError);
  }
  i++;
}
}
return;
}
```

## Services and Distributed Transaction Processing in Oracle Real Application Clusters

An XA transaction can span Oracle RAC instances by default, allowing any application that uses XA to take full advantage of the Oracle RAC environment to enhance the availability and scalability of the application.

This feature is controlled through the `GLOBAL_TXN_PROCESSES` initialization parameter, which is set to `1` by default. This parameter specifies the initial number of GTX*n* background processes for each Oracle RAC instance. Keep this parameter at its default value clusterwide to allow distributed transactions to span more than one Oracle RAC instance.

This allows the units of work performed across these Oracle RAC instances to share resources and act as a single transaction (that is, the units of work are tightly coupled). It also allows 2PC requests to be sent to any node in the cluster. Tightly coupled XA



ORCL01972420

transactions no longer require the special type of singleton services (that is, Oracle Distributed Transaction Processing (DTP) services) to be deployed on Oracle RAC database. XA transactions are transparently supported on Oracle RAC databases with any type of services configuration.

> **See Also:** *Oracle Database Advanced Application Developer's Guide* for complete information about using Oracle XA with Oracle RAC, and *Oracle Database Reference* for information about the `GLOBAL_TXN_PROCESSES` initialization parameter

To provide improved application performance with distributed transaction processing in Oracle RAC, you may want to take advantage of the specialized service referred to as a DTP Service. Using DTP services, you can direct all branches of a distributed transaction to a single instance in the cluster. To load balance across the cluster, it is better to have several groups of smaller application servers with each group directing its transactions to a single service, or set of services, than to have one or two larger application servers.

In addition, connection pools at the application server tier that load balance across multiple connections to an Oracle RAC database can use this method to ensure that all tightly-coupled branches of a global distributed transaction run on only one Oracle RAC instance. This is also true in distributed transaction environments using protocols such as X/Open Distributed Transaction Processing (DTP) or the Microsoft Distributed Transaction Coordinator (DTC).

To enhance the performance of distributed transactions, you can use services to manage DTP environments. By defining the DTP property of a service, the service is guaranteed to run on one instance at a time in an Oracle RAC database. All global distributed transactions performed through the DTP service are ensured to have their tightly-coupled branches running on a single Oracle RAC instance. This has the following benefits:

- The changes are available locally within one Oracle RAC instance when tightly coupled branches need information about changes made by each other
- Relocation and failover of services are fully supported for DTP
- By using more DTP services than there are Oracle RAC instances, Oracle can balance the load by services across all of the Oracle RAC database instances

To leverage all of the instances in a cluster, create one or more DTP services for each Oracle RAC instance that hosts distributed transactions. Choose one DTP service for one distributed transaction. Choose different DTP services for different distributed transactions to balance the workload among the Oracle RAC database instances. Because all of the branches of a distributed transaction are on one instance, you can leverage all of the instances to balance the load of many DTP transactions through multiple singleton services, thereby maximizing application throughput.

An external transaction manager, such as OraMTS, coordinates DTP/XA transactions. However, an internal Oracle transaction manager coordinates distributed SQL transactions. Both DTP/XA and distributed SQL transactions must use the DTP service in Oracle RAC.

If you add or delete nodes from your cluster database, then you may need to identify and relocate services that you are using for DTP transactions to ensure that you maintain optimum performance levels.



ORCL01972421

**See Also:** *Oracle Database Advanced Application Developer's Guide* for more information about transaction branch management in Oracle RAC

## Enabling Distributed Transaction Processing for Services

For services that you are going to use for distributed transaction processing, create the service using Enterprise Manager, or SRVCTL and define only one instance as the preferred instance. You can have as many AVAILABLE instances as you want. For example, the following SRVCTL command creates a singleton service for database crm, xa_01.service.us.oracle.com, whose preferred instance is RAC01:

```
srvctl add service -d crm -s xa_01.service.us.oracle.com -r RAC01 -a RAC02, RAC03
```

Then mark the service for distributed transaction processing by setting the DTP parameter to TRUE; the default is FALSE. Enterprise Manager enables you to set this parameter on the Cluster Managed Database Services: Create Service or Modify Service page. You can also use the DBMS_SERVICE package to modify the DTP property of the singleton service, as follows:

```
EXECUTE DBMS_SERVICE.MODIFY_SERVICE(service_name =>'xa_01.service.us.oracle.com',
DTP=>TRUE);
```

If, for example, RAC01 (that provides service XA_01) fails, then the singleton service that it provided fails over to one of the other instances, such as RAC02 or RAC03.

If services migrate to other instances after the cold-start of the Oracle RAC database, then you might need to force the relocation of the service to evenly re-balance the load on all of the available hardware. Use data from the GV$ACTIVE_SERVICES view to determine whether to do this.

## Administering Services

When you create and administer services, you are dividing the work that your database performs into manageable units. The goal of using services is to achieve optimal utilization of your database infrastructure. You can create and deploy services based on business requirements and Oracle Database can measure the performance for each service. You can define both the application modules within a service as well as the individual actions for a module and monitor thresholds for these actions. This enables you to manage workloads to deliver capacity on demand.

When you create new services for your database, you should define each service's automatic workload management characteristics. The characteristics of a service include:

- A unique global name to identify the service.
- An Oracle Net Services name that a client uses to connect to the service.
- The preferred instances where you want the service to be enabled in a normal running environment.
- The available instances where the service can be enabled if one of the preferred instances fails.
- A service goal that determines whether connections are made to the service based on best service quality (how efficiently a single transaction completes) or best throughput (how efficiently a complete job or long-running query completes), as determined by the load balancing advisory.



ORCL01972422

- An indicator that determines whether the service is used for distributed transactions.
- An indicator that determines whether Oracle RAC high availability events are sent to OCI and ODP.NET clients that have registered to receive them through Advanced Queuing.

  **See Also:** "Enabling Oracle Call Interface Clients for Fast Connection Failover" on page 4-15 for more details

- The characteristics of session failovers such as whether failovers occur, whether sessions can use pre-existing connections on the failover instance, and whether failed over sessions can continue to process interrupted queries. The service definition can also define the number of times that a failed session attempts to reconnect to the service and how long it should wait between reconnection attempts. The service definition can also include a connection load balancing goal that informs the listener how to balance connection requests across the instances that provide the service.
- The method for load balancing connections for each service. This method is used by the listener when Oracle creates connections. Connections are classified as `LONG` (such as connection pools and SQL*FORMS) which tells the listener to use session based, or `SHORT` which tells the listener to use CPU based. If load balancing advisory is enabled, its information is used to balance connections otherwise CPU utilization is used.

In addition to creating services, you may need to:

- Delete a service. You can delete services that you created. However, you cannot delete or modify the properties of the default database service that Oracle Database created.
- Check the status of a service. A service can be assigned different roles among the available instances. In a complex database with many services, you may not remember the details of every service. Therefore, you may need to check the status on an instance or service basis. For example, you may need to know the status of a service for a particular instance before you make modifications to that instance or to the Oracle home from which it runs.
- Start or stop a service for a database or an instance. A service must be started before it can be used for client connections to that instance. If you shutdown your database, for example, by running the Server Control (`SRVCTL`) command `srvctl stop database -d` *`database_name`* where *`database_name`* is the name of the database that you want to stop, then Oracle Database stops all services to that database. You must manually restart the services when you start the database.
- Map a Service to a Consumer Group—Oracle Database can automatically map services to Resource Manager Consumer groups to limit the amount of resources that services can use in an instance. You must create the consumer group and then map the service to the consumer group.

  **See Also:** *Oracle Database PL/SQL Packages and Types Reference* for information about the `DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING_PRI` procedure

- Enable or disable a service for a database or an instance. By default, Oracle Clusterware attempts to restart a service automatically after failures. You can prevent this behavior by disabling a service. Disabling a service is useful when



ORCL01972423

you need to perform database or instance maintenance, for example, if you are performing an upgrade and you want to prevent connection requests.

- Relocate a service to a different instance. You can move a service from one instance to another instance to re-balance workloads, for example, after adding or deleting cluster nodes.
- If you execute a SQL statement in parallel, the parallel processes only run on the instances that offer the service with which you originally connected to the database. This is the default behavior. This does not affect other parallel operations such as parallel recovery or the processing of `GV$` queries. To override this behavior, set a value for the `PARALLEL_INSTANCE_GROUPS` initialization parameter.

**Notes:**

- When you use services, do not set a value for the `SERVICE_NAMES` parameter; Oracle controls the setting for this parameter for the services that you create and for the default database service. The service features that this chapter describes are not directly related to the features that Oracle provides when you set `SERVICE_NAMES`. In addition, setting a value for this parameter may override some of the benefits of using services.
- Using service names to access a queue provides location transparency for the queue in an Oracle RAC database.
- If you specify a service using the `DISPATCHERS` initialization parameter, it overrides any service in the `SERVICE_NAMES` parameter, and cannot be managed. (For example, stopping the service with a `SRVCTL` command will not stop users connecting with the service.)

# Administering Services with Enterprise Manager, PL/SQL, and SRVCTL

You can create and administer services with Enterprise Manager and the `DBMS_SERVICE` PL/SQL package, and you can perform most service administration tasks with the `SRVCTL` utility. The following sections describe how to perform service-related tasks using these tools:

- Administering Services with Enterprise Manager
- Administering Services with the PL/SQL DBMS_SERVICE Package
- Administering Services with SRVCTL

## Administering Services with Enterprise Manager

The Cluster Managed Database Services page is the master page for beginning all tasks related to services. To access this page, go to the Cluster Database Availability page, then click **Cluster Managed Database Services** in the Services section. You can use this page and drill down from this page to:

- View a list of services for the cluster
- View the instances on which each service is currently running
- View the status for each service
- Create or edit a service



ORCL01972424

- Start or stop a service
- Enable or disable a service
- Perform instance-level tasks for a service
- Delete a service

> **See Also:** *Oracle Enterprise Manager Concepts* for more information about administering services with Enterprise Manager

### Service-Related Tasks That You Can Perform with Enterprise Manager

You can perform service-related tasks as described for the following Enterprise Manager pages:

- Cluster Managed Database Services Page
- Cluster Managed Database Services Detail Page
- Create Services Page

### Cluster Managed Database Services Page

The Cluster Managed Database Services page enables you to:

- View a list of services for the cluster, the instances on which each service is currently running, and the status for each service
- Start or stop a service, or enable or disable a service
- Access the Create Service and Edit Service pages
- Access the Services Detail page to perform instance-level tasks for a service
- Test the connection for a service

### Cluster Managed Database Services Detail Page

The Cluster Managed Database Services Detail page enables you to:

- View the status of a service on all of its preferred and available instances; the status can be `Running`, `Stopped`, or `Disabled`
- Stop or start a service for an instance of a cluster database
- Disable or enable a service for an instance of a cluster database
- Relocate a service to manually re-balance the services load

### Create Services Page

The Create Services page enables you to:

- Create the service with name, high availability and performance attributes
- Select the desired service policy for each instance configured for the cluster database
- Select the desired service properties, such as the TAF policy. (This configures client-side TAF. To set the service TAF policy, use the `DBMS_SERVICE` package.) You can also set the notification properties, load balancing goals, alert levels, and resource management properties

> **See Also:** *Oracle Database Net Services Administrator's Guide* for more information about configuring TAF



ORCL01972425

### Accessing the Enterprise Manager Services Pages

To access the Cluster Managed Database Services page and detail pages for service instances:

1. From the Cluster Database Home page, click the **Availability** tab.
2. From the Cluster Database Availability page, under the Services heading in the High Availability options list, click **Cluster Managed Database Services**. The Cluster and Database Login page appears.
3. Enter credentials for the database and for the cluster that hosts the cluster database and click **Continue**. The Cluster Managed Database Services page appears and displays services that are available on the cluster database instances. For information about performing tasks on this page, see the online help for this page.

> **Note:** You must have `SYSDBA` credentials to access a cluster database. Cluster Managed Database Services does not permit you to connect as anything other than `SYSDBA`.

4. To manage a service at the instance level, either click a service name or select a service name, then select **Manage** from the Actions drop-down list and click **Go**. The Cluster Managed Database Services Detail page for the service appears. For information about performing tasks on this page, see the online help for this page.

To access the Relocate page:

1. Perform steps 1 - 3 from the previous procedure set.
2. From the Cluster Managed Database Services page, either click a service name or select a service name, then select **Manage** from the Actions drop-down list and click **Go**. The Cluster Managed Database Services Detail page for the service appears.
3. Click **Relocate**. The Relocate Service from Instance page appears, where you can perform manual load balancing of services. For information about performing tasks on this page, see the online help for this page.

## Administering Services with the PL/SQL DBMS_SERVICE Package

You can manage services with the PL/SQL `DBMS_SERVICE` package procedures described in Table 4–4.

> **Note:** Oracle recommends that you use Enterprise Manager to create services for Oracle RAC environments.



ORCL01972426

*Table 4–4 DBMS_SERVICE Package Procedures for Administering Services*

| Procedure | Description |
|---|---|
| CREATE_SERVICE | Adds a new service to the Oracle RAC database. In the CREATE_SERVICE syntax:<br>▪ service_name is the unique, global service name<br>▪ network_name is the TNS name for connections to the service<br>▪ goal sets the automatic workload management goal directive to service quality or throughput<br>▪ dtp is the distributed transaction flag<br>▪ aq_ha_notification is the flag to send Oracle RAC high availability events to registered OCI clients<br>▪ failover_method is the TAF failover method for the service<br>▪ failover_type is the TAF failover method for the service<br>▪ failover_retries is the TAF connection retry count<br>▪ failover_delay is the wait time between TAF connection retries<br>▪ clb_goal sets the runtime connection load balancing goal<br>When using services with Oracle RAC, use Enterprise Manager or SRVCTL commands to add the high availability properties, such as the PREFERRED and AVAILABLE placement. |
| MODIFY_SERVICE | Changes one or more service characteristics. The parameters for the MODIFY_SERVICE procedure are the same as those for the CREATE_SERVICE procedure. |
| DELETE_SERVICE | Drops an existing service. In the DELETE_SERVICE procedure, service_name is the name of the service to be dropped. |
| START_SERVICE | Starts a service on the connected instance, the named instance, or all instances. In the START_SERVICE syntax SERVICE_NAME is the name of the service to be started and INSTANCE_NAME can be NULL (to start the service on the connected instance), the name of an instance (to start the service on that instance), or the package constant DBMS_SERVICE.ALL_INSTANCES to start the service on all instances where it is defined. |
| STOP_SERVICE | Stops a service on the connected instance, the named instance, or all instances. The parameters for the STOP_SERVICE procedure are the same as those for the START_SERVICE procedure. |
| DISCONNECT_SESSION | Terminates all sessions on the connected instance, or the instance where the PL/SQL procedure is run. In the DISCONNECT_SESSION procedure, service_name is the name of the service for which connections are to be dropped. |

**See Also:** *Oracle Database PL/SQL Packages and Types Reference* for more information about the CREATE_SERVICE procedure syntax and for information about other procedures mentioned in this section

Also, see the *Oracle Database Net Services Administrator's Guide* for more information about configuring TAF

## Administering Services with SRVCTL

When you create a service with SRVCTL, you must start it with a separate SRVCTL command. However, you may later need to manually stop or restart the service. You may also need to disable the service to prevent automatic restarts, to manually relocate



ORCL01972427

the service, or obtain status information about the service. The following sections explain how to use SRVCTL to perform the following administrative tasks:

- Creating Services with SRVCTL
- Starting and Stopping Services with SRVCTL
- Enabling and Disabling Services with SRVCTL
- Relocating Services with SRVCTL
- Obtaining the Statuses of Services with SRVCTL
- Obtaining the Configuration of Services with SRVCTL

> **See Also:** Appendix A, "Server Control Utility Reference" for more information about SRVCTL and the other SRVCTL commands that you can use to manage services

### Creating Services with SRVCTL

To create a service with SRVCTL, enter a command from the command line using the following syntax:

```
srvctl add service -d database_unique_name -s service_name -r preferred_list
[-a available_list] [-P TAF_policy]
```

### Starting and Stopping Services with SRVCTL

Enter the following SRVCTL syntax from the command line:

```
srvctl start service -d database_unique_name [-s service_name_list] [-i inst_name]
[-o start_options]

srvctl stop service -d database_unique_name -s service_name_list [-i inst_name]
[-o start_options]
```

### Enabling and Disabling Services with SRVCTL

Use the following SRVCTL syntax from the command line to enable and disable services:

```
srvctl enable service -d database_unique_name -s service_name_list [-i inst_name]
srvctl disable service -d database_unique_name -s service_name_list [-i inst_name]
```

### Relocating Services with SRVCTL

Run the `srvctl relocate service` command from the command line to relocate a service. For example, the following command relocates the `crm` service from instance apps1 to instance apps3:

```
srvctl relocate service -d apps -s crm -i apps1 -t apps3
```

### Obtaining the Statuses of Services with SRVCTL

Run the `srvctl relocate service` command from the command line to obtain the status of a service. For example, the following command returns the status of the `crm` service that is running on the crm database:

```
srvctl status service -d apps -s crm
```



ORCL01972428

### Obtaining the Configuration of Services with SRVCTL

Run the `srvctl relocate service` command from the command line to obtain the high availability configuration of a service. For example, the following command returns the configuration of the `crm` service that is running on the `crm` database:

```
srvctl config service -d apps -s crm -a
```

> **See Also:** Appendix A, "Server Control Utility Reference" for information about other administrative tasks that you can perform with `SRVCTL`

## Measuring Performance by Service Using the Automatic Workload Repository

Services add a new dimension for performance tuning. With services, workloads are visible and measurable and resource consumption and wait times are attributable by application. Tuning by using 'service and SQL' replaces tuning by 'session and SQL' in the majority of systems where all sessions are anonymous and shared.

The Automatic Workload Repository (AWR) maintains performance statistics that include information about response time, throughput, resource consumption, and wait events for all services and work that a database performs. Oracle also maintains metrics, statistics, wait events, wait classes, and SQL-level traces for services. You can optionally augment these statistics by defining modules within your application to monitor certain statistics. You can also define the actions within those modules that business critical transactions should execute in response to particular statistical values. Enable module and action monitoring using the `DBMS_MONTIOR` PL/SQL package as follows:

```
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(SERVICE_NAME => 'ERP', MODULE_NAME=>
'PAYROLL', ACTION_NAME => 'EXCEPTIONS PAY');
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(SERVICE_NAME => 'ERP', MODULE_NAME=>
'PAYROLL', ACTION_NAME => NULL);
```

Use the `DBA_ENABLED_AGGREGATIONS` view to verify that you have enabled monitoring.

Statistics aggregation and tracing by service are global in scope for Oracle RAC databases. In addition, they are persistent across instance restarts and service relocations for both Oracle RAC and single-instance Oracle databases.

The service, module, and action names are visible in `V$SESSION`, `V$ACTIVE_SESSION_HISTORY`, and `V$SQL` views. The call times and performance statistics are visible in `V$SERVICE_STATS`, `V$SERVICE_EVENTS`, `V$SERVICE_WAIT_CLASSES`, `V$SERVICEMETRIC`, and `V$SERVICEMETRIC_HISTORY`. When you enable statistics collection for an important transaction, you can see the call speed for each service, module, and action name at each database instance using the `V$SERV_MOD_ACT_STATS` view.

The following sample SQL*Plus script provides service quality statistics every five seconds. You can use these service quality statistics to monitor the quality of a service, to direct work, and to balance services across Oracle RAC instances:

```
SET PAGESIZE 60 COLSEP '|' NUMWIDTH 8 LINESIZE 132 VERIFY OFF FEEDBACK OFF
COLUMN service_name FORMAT A20 TRUNCATED HEADING 'Service'
COLUMN begin_time HEADING 'Begin Time' FORMAT A10
COLUMN end_time HEADING 'End Time' FORMAT A10
COLUMN instance_name HEADING 'Instance' FORMAT A10
COLUMN service_time HEADING 'Service Time|mSec/Call' FORMAT 999999999
```



ORCL01972429

```
COLUMN throughput HEADING 'Calls/sec'FORMAT 99.99
BREAK ON service_name SKIP 1
SELECT
    service_name
  , TO_CHAR(begin_time, 'HH:MI:SS') begin_time
  , TO_CHAR(end_time, 'HH:MI:SS') end_time
  , instance_name
  , elapsedpercall  service_time
  ,  callspersec  throughput
FROM
    gv$instance i
  , gv$active_services s
  , gv$servicemetric m
WHERE s.inst_id = m.inst_id
  AND s.name_hash = m.service_name_hash
  AND i.inst_id = m.inst_id
  AND m.group_id = 10
ORDER BY
   service_name
 , i.inst_id
 , begin_time ;
```

## Service Thresholds and Alerts

Service level thresholds enable you to compare achieved service levels against accepted minimum required levels. This provides accountability with respect to the delivery or the failure to deliver an agreed service level. The end goal is a predictable system that achieves service levels. There is no requirement to perform as fast as possible with minimum resource consumption; the requirement is to meet the *quality* of service.

You can explicitly specify two performance thresholds for each service: the response time for calls, or `SERVICE_ELAPSED_TIME`, and the CPU time for calls, or `SERVICE_CPU_TIME`. The response time goal indicates that the elapsed time should not exceed a certain value, and the response time represents wall clock time. Response time is a fundamental measure that reflects all delays and faults that might be blocking the call from running on behalf of the user. Response time can also indicate differences in node power across the nodes of an Oracle RAC database.

The service time and CPU time are calculated as the moving average of the elapsed, server-side call time. The AWR monitors the service time and CPU time and publishes AWR alerts when the performance exceeds the thresholds. You can then respond to these alerts by changing the priority of a job, stopping overloaded processes, or by relocating, expanding, shrinking, starting or stopping a service. This permits you to maintain service availability despite changes in demand.

### Example of Services and Thresholds Alerts

To check the thresholds for the payroll service, use the AWR report. You should record output from the report over several successive intervals during which time the system is running optimally. For example, assume that for an e-mail server, the AWR report runs each Monday during the peak usage times of 10:00 a.m. to 2:00 p.m. The AWR report would contain the response time, or DB time, and the CPU consumption time, or CPU time, for calls for each service. The AWR report would also provide a breakdown of the work done and the wait times that are contributing to the response times.



ORCL01972430

Using DBMS_MONITOR, set a warning threshold for the payroll service at 0.5 seconds and a critical threshold for the payroll service at 0.75 seconds. You must set these thresholds at all instances within an Oracle RAC database. You can schedule actions using Enterprise Manager jobs for alerts, or you can schedule actions to occur programmatically when the alert is received. In this example, thresholds are added for the servall service and set as follows:

```
EXECUTE DBMS_SERVER_ALERT.SET_THRESHOLD(
METRICS_ID => DBMS_SERVER_ALERT.ELAPSED_TIME_PER_CALL
, warning_operator => DBMS_SERVER_ALERT.OPERATOR_GE
, warning_value => '500000'
, critical_operator => DBMS_SERVER_ALERT.OPERATOR_GE
, critical_value => '750000'
, observation_period => 30
, consecutive_occurrences => 5
, instance_name => NULL
, object_type => DBMS_SERVER_ALERT.OBJECT_TYPE_SERVICE
, object_name => 'servall');
```

Verify the threshold configuration using the following SELECT statement:

```
SELECT METRICS_NAME, INSTANCE_NAME, WARNING_VALUE, CRITICAL_VALUE,
OBSERVATION_PERIOD FROM dba_thresholds ;
```

## Enable Service, Module, and Action Monitoring

You can enable performance data tracing for important modules and actions within each service. The performance statistics are available in the V$SERV_MOD_ACT_STATS view. As an example, set the following:

- Under the ERP service, enable monitoring for the exceptions pay action in the payroll module.
- Under the ERP service, enable monitoring for the all actions in the payroll module.
- Under the HOT_BATCH service, enable monitoring for all actions in the posting module.

```
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(service_name => 'erp', module_name=>
 'payroll', action_name => 'exceptions pay');
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(service_name => 'erp', module_name=>
 'payroll', action_name => NULL);
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(service_name => 'hot_batch',
module_name =>'posting', action_name => NULL);
```

Verify the enabled service, module, action configuration with the following SELECT statement:

```
COLUMN AGGREGATION_TYPE FORMAT A21 TRUNCATED HEADING 'AGGREGATION'
COLUMN PRIMARY_ID FORMAT A20 TRUNCATED HEADING 'SERVICE'
COLUMN QUALIFIER_ID1 FORMAT A20 TRUNCATED HEADING 'MODULE'
COLUMN QUALIFIER_ID2 FORMAT A20 TRUNCATED HEADING 'ACTION'
SELECT * FROM DBA_ENABLED_AGGREGATIONS ;
```

The output might appear as follows:

```
AGGREGATION            SERVICE              MODULE      ACTION
------------           --------------------  ----------  -------------
SERVICE_MODULE_ACTION  ERP                   PAYROLL     EXCEPTIONS PAY
SERVICE_MODULE_ACTION  ERP                   PAYROLL
SERVICE_MODULE_ACTION  HOT_BATCH             POSTING
```



ORCL01972431

ORCL01972432