

Oracle® Web Cache

Administration and Deployment Guide

Release 1.0.2.3

March 2, 2001

Part No. A86722-03

ORACLE®

Part No. A86722-03

Copyright © 1999, 2001 Oracle Corporation. All rights reserved.

Primary Author: Deborah Steiner

Contributors: Henry Abrecht, Jose Alvarez, Steve Andrew, Jesse Anton, Pierre Baudin, Omar Bellal, Pramodini Gattu, Larry Jacobs, Xiang Liu, Shehzaad Nakhoda, Marcin Porwit, Jean Zeng, Tie Zhong

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 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 Oracle*9i* and Oracle*8i* are trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

Regular Expression Rights and License Notice The Windows NT release 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, 2001, 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.
5. Redistributions of any form whatsoever must retain the following acknowledgment:
"This product includes PHP, freely available from <http://www.php.net/>."
6. The software incorporates the Zend Engine, a product of Zend Technologies, Ltd. ("Zend"). The Zend Engine is licensed to the PHP Association (pursuant to a grant from Zend that can be found at <http://www.php.net/license/ZendGrant/>) for distribution to you under this license agreement, only as a part of PHP. In the event that you separate the Zend Engine (or any portion thereof)

from the rest of the software, or modify the Zend Engine, or any portion thereof, your use of the separated or modified Zend Engine software shall not be governed by this license, and instead shall be governed by the license set forth at <http://www.zend.com/license/ZendLicense/>.

THIS SOFTWARE IS PROVIDED BY THE PHP DEVELOPMENT TEAM "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE PHP DEVELOPMENT TEAM OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the PHP Group.

The PHP Group can be contacted through email at group@php.net.

For more information on the PHP Group and the PHP project, please see <http://www.php.net>.

Contents

Send Us Your Comments	xv
Preface.....	xvii
1 Introduction to Oracle Web Cache	
What is the Big Picture for Caching?	1-2
Oracle's Solution to Web Site Performance Issues.....	1-2
How Web Caching Works	1-3
Benefits of Web Caching.....	1-5
Features of Oracle Web Cache	1-7
Static and Dynamically Generated Content Caching.....	1-7
Cache Invalidation.....	1-8
Performance Assurance	1-8
Surge Protection of Application Web Servers.....	1-9
Load Balancing of Application Web Servers.....	1-9
Backend Failover	1-10
Application Web Server Binding	1-12
Security Features.....	1-14
Administration.....	1-14
Compression.....	1-14
2 Oracle Web Cache Concepts	
Populating Oracle Web Cache	2-2
Cache Freshness and Performance Assurance	2-2

Caching Dynamically Generated Content	2-5
Multiple Versions of the Same URL.....	2-6
Personalized Attributes	2-9
Session Information.....	2-11
Session Tracking	2-11
Session-Encoded URLs	2-12
3 Deploying Oracle Web Cache	
Caching Content for One Application Web Server.....	3-2
Forwarding HTTPS Requests	3-4
Load Balancing Requests Among Application Web Servers	3-6
Accelerating Portions of a Web Site	3-8
Using Oracle Web Cache Servers in a Failover Pair.....	3-10
Working with Firewalls	3-11
Deploying Oracle Web Cache Servers in a Distributed Network.....	3-14
4 Configuration and Administration Tools Overview	
Oracle Web Cache Manager	4-2
Starting Oracle Web Cache Manager	4-2
Navigating Oracle Web Cache.....	4-3
Apply Changes and Cancel Changes Buttons.....	4-4
Status Messages	4-4
Navigator Pane	4-5
Right Pane.....	4-6
webcachectl Utility	4-6
Configuration and Administration Tasks at a Glance	4-7
5 Initial Setup and Configuration	
Task 1: Start Oracle Web Cache	5-2
Task 2: Modify Security Settings	5-2
Task 3: Set Resource Limits	5-5
Task 4: Specify Web Site Settings	5-7
Task 5: Specify Caching Rules.....	5-10
Task 6: Restart Oracle Web Cache.....	5-10

6 Creating Rules for Cached Content

Cacheability Rules Overview	6-2
Cacheability Rule Syntax.....	6-3
Default Cacheability Rules.....	6-4
Configuring Cacheability Rules	6-5
Cacheability Rule Example	6-10
Configuring Expiration Rules	6-12
Configuring Rules for Multiple-Version URLs Containing Cookies	6-14
Configuring Rules for Multiple-Version URLs Containing HTTP Request Headers	6-15
Configuring Rules for Personalized Pages	6-16
Example: Personalized Page Configuration	6-19
Configuring Rules for Pages with Session Tracking	6-24

7 Configuration Considerations for Web Sites with Multiple Application Web Servers

Configuring Load Balancing	7-2
Binding a Session to an Application Web Server	7-3

8 Administering Oracle Web Cache

Starting and Stopping Oracle Web Cache	8-2
Invalidate Documents in the Cache	8-4
Setting the Invalid Port Number	8-4
Sending Invalid Messages	8-5
Manual Invalid Using Telnet.....	8-6
Manual Invalid Using Oracle Web Cache Manager.....	8-11
Automatic Invalid Using Database Triggers.....	8-15
Automatic Invalid Using Scripts.....	8-15
Automatic Invalid Using Applications	8-15
Invalidation Examples	8-16
Example: Invalidating One Document.....	8-16
Example: Invalidating a Subtree of Documents	8-17
Example: Invalidating All Documents for a Web Site	8-18

Evaluating Event Logs.....	8-19
Format of the Event Log File.....	8-19
Event Log Examples.....	8-19
Example: Event Log with Startup Entries.....	8-19
Example: Event Log with Unsuccessful Startup Entries	8-20
Example: Event Log with an Invalidation Entry	8-20
Example: Event Log with an Invalidation Message Error.....	8-20
Example: Event Log with Shutdown Entries	8-21
Finding Errors in the Event Log	8-21
Configuring Event Logs.....	8-21
Evaluating Access Logs.....	8-22
Format of the Access Log File.....	8-23
Access Log Examples	8-24
Example: Access Log with Reload Entries.....	8-25
Example: Access Log with Wrong Path Entry	8-26
Example: Access Log with Status Code 404 Entry.....	8-26
Example: Access Log with Status Code 304 Entry.....	8-26
Configuring Access Logs	8-27

9 Monitoring Performance

Setting the Statistics Monitoring Port Number	9-2
Monitoring Overall Cache Health	9-3
Gathering Oracle Web Cache Performance Statistics	9-5
Gathering Application Web Server Performance Statistics.....	9-6

A Oracle Web Cache Directory Structure

B Oracle Web Cache Default Settings

C Invalidation Document Type Declaration

Invalidation Request DTD.....	C-2
Invalidation Response DTD	C-5

D Event Log Messages

Information Events	D-2
Warning Events	D-4
Error Events	D-5

E Troubleshooting Oracle Web Cache Configuration

Startup Failures	E-2
Port Conflicts.....	E-2
Cache Memory	E-4
Privileged Ports.....	E-5
Application Web Server Capacity	E-7
Wrong or Older Cached Content	E-7
Configuration Changes Made in Oracle Web Cache Manager	E-8

Glossary

Index

List of Figures

1–1	Oracle Web Cache Architecture	1-3
1–2	Web Server Acceleration	1-5
1–3	Load Balancing	1-10
1–4	Failover	1-11
1–5	Application Web Server Binding	1-13
2–1	Performance Assurance Heuristics Graph	2-4
2–2	Multiple-Version URL	2-6
2–3	Page with a Personalized Attribute	2-10
2–4	Session-Encoded URLs	2-12
3–1	Oracle Web Cache On Same Computer As the Application Web Server	3-2
3–2	Oracle Web Cache On a Different Computer From the Application Web Server	3-3
3–3	Using an L4 Switch to Forward HTTPS Requests	3-5
3–4	Load Balancing with Oracle Web Cache.....	3-6
3–5	Load Balancing with a Third-Party Load Balancer	3-7
3–6	Accelerating Portions of a Web Site.....	3-9
3–7	Configuring Multiple Oracle Web Caches as a Failover Pair	3-10
3–8	Configuring Oracle Web Cache Inside a Firewall	3-12
3–9	Configuring Oracle Web Cache Outside a Firewall.....	3-13
3–10	Distributed Caching.....	3-15
3–11	Centralizing the Data Source	3-17
4–1	Oracle Web Cache Configuration Interface.....	4-3
4–2	Cacheability Rules Property Sheet.....	4-6
6–1	Cacheability Rules Example	6-10
6–2	monthly.htm.....	6-19
6–3	Edit/Create Session/Personalized Attribute Definition Dialog Box	6-20
6–4	Add Session Related Caching Rule Dialog Box.....	6-21
6–5	Create Cacheability Rule Dialog Box.....	6-22
6–6	monthly.htm When Cached.....	6-23
8–1	Invalidation	8-5
8–2	Event Log with Successful Startup Entries.....	8-19
8–3	Event Log with an Unsuccessful Startup.....	8-20
8–4	Event Log with an Invalidation Entry.....	8-20
8–5	Event Log with an Invalidation Message Error	8-20
8–6	Event Log with Shutdown Entries.....	8-21
8–7	Access Log	8-24
8–8	Access Log with Reload Entries	8-25
8–9	Access Log with Wrong Path Entry.....	8-26
8–10	Access Log with HTTP Status Code 404 Entry	8-26
8–11	Access Log with HTTP Status Code 304 Entry	8-26
C–1	Invalidation Request DTD	C-2

C-2	Invalidation Response DTD	C-5
E-1	Event Log with Port Conflict Messages	E-2
E-2	Event Log with Administration Port Conflict Messages	E-3
E-3	webcache.xml File.....	E-3
E-4	Event Log with Cache Memory Messages.....	E-4

List of Tables

2–1	Multiple-Version URL with Different Cookie Values.....	2-7
2–2	HTTP Request Headers	2-8
4–1	Oracle Web Cache Manager Status Messages.....	4-4
4–2	Common Administrative Tasks for Oracle Web Cache.....	4-7
6–1	Regular Expression Examples	6-3
6–2	Regular Expression Examples	6-4
6–3	Regular Expression Examples	6-11
8–1	Invalidation Message Syntax.....	8-7
8–2	Invalidation Response Syntax	8-10
8–3	XLF Fields for Access Logs	8-23
9–1	Oracle Web Cache Health Monitor Statistics	9-3
9–2	Oracle Web Cache Statistics	9-5
9–3	Application Web Server Statistics.....	9-7
A–1	Oracle Web Cache Directory Structure	A-1
B–1	Oracle Web Cache Default Settings	B-1
C–1	Invalidation Request DTD Elements and Attributes	C-3
C–2	Invalidation Response DTD Elements and Attributes.....	C-5
D–1	Information Events.....	D-2
D–2	Warning Events	D-4
D–3	Error Events.....	D-5

Send Us Your Comments

Oracle Web Cache Administration and Deployment Guide, Release 1.0.2.3

Part No. A86722-03

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: infodev_us@oracle.com
- FAX: (650) 506-7227 Attn: Information Development
- Postal service:
Oracle Corporation
Information Development Documentation Manager
500 Oracle Parkway, Mailstop 4OP11
Redwood Shores, CA 94065
USA

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Oracle Web Cache Administration and Deployment Guide describes how to use Oracle Web Cache to cache both static and dynamically generated content from one or more application Web servers.

This preface contains these topics:

- [Audience](#)
- [Organization](#)
- [Related Documentation](#)
- [Conventions](#)

Audience

Oracle Web Cache Administration and Deployment Guide is intended for Web site administrators who perform the following tasks:

- Web site administration
- Application Web server administration
- **Domain Name System (DNS)** administration

To use this document, you need to be familiar with release 1.0 and 1.1 of the **HTTP protocol**, as well as application Web server and DNS administration.

Organization

This document contains:

Chapter 1, "Introduction to Oracle Web Cache"

This chapter introduces the architecture, benefits, and main features of Oracle Web Cache.

Chapter 2, "Oracle Web Cache Concepts"

This chapter explains how Oracle Web Cache is populated with content, how that content maintains consistency, and how dynamically generated content is cached.

Chapter 3, "Deploying Oracle Web Cache"

This chapter presents several scenarios for deploying Oracle Web Cache.

Chapter 4, "Configuration and Administration Tools Overview"

This chapter introduces the various administration tools of Oracle Web Cache. It discusses the main administration application and tells you how to launch it and navigate through it.

Chapter 5, "Initial Setup and Configuration"

This chapter describes the steps to initially configure Oracle Web Cache to begin caching application Web server content after installation.

Chapter 6, "Creating Rules for Cached Content"

This chapter explains how to configure cacheability rules.

Chapter 7, "Configuration Considerations for Web Sites with Multiple Application Web Servers"

This chapter describes load balancing, failover, and session binding configuration options available for deployments with two more application Web servers.

Chapter 8, "Administering Oracle Web Cache"

This chapter describes how to start and stop Oracle Web Cache, invalidate documents in the cache, and evaluate event and access log files.

Chapter 9, "Monitoring Performance"

This chapter describes how to gather performance statistics and how to interpret them.

Appendix A, "Oracle Web Cache Directory Structure"

This appendix describes the installed Oracle Web Cache directory structure.

Appendix B, "Oracle Web Cache Default Settings"

This appendix describes the default settings for Oracle Web Cache.

Appendix C, "Invalidation Document Type Declaration"

This appendix describes the Document Type Declaration (DTD), or grammar, of invalidation requests and responses.

Appendix D, "Event Log Messages"

This appendix describes the most common event log messages.

Appendix E, "Troubleshooting Oracle Web Cache Configuration"

This appendix describes common configuration problems and debugging techniques for resolving them.

Glossary

Related Documentation

For more information, see these Oracle resources:

- The Oracle Internet Application Server documentation set, especially:
 - *Oracle Internet Application Server 8i Overview Guide*
 - *Oracle Internet Application Server 8i Oracle HTTP Server powered by Apache Performance Guide*
- *PL/SQL User's Guide and Reference*

In North America, printed documentation is available for sale in the Oracle Store at

<http://oraclestore.oracle.com/>

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

<http://www.oraclebookshop.com/>

Other customers can contact their Oracle representative to purchase printed documentation.

To download patches, please visit the Oracle Technology Network at

<http://otn.oracle.com/products/ias>

For additional information, see:

- http://www.cs.utah.edu/dept/old/texinfo/regex/regex_toc.html for **regular expression** syntax
- <http://www.ietf.org/> for information about the **Open Systems Interconnection (OSI)**
- <http://www.cookiecentral.com/> for further information about **cookies**
- <http://www.ietf.org/rfc/rfc2616.txt> for further information about the HTTP protocol

Conventions

This section describes the conventions used in the text and code examples of the Oracle Internet Application Server documentation set. It describes:

- [Conventions in Text](#)
- [Conventions in Code Examples](#)

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	The C datatypes such as ub4 , sword , or OCINumber are valid. When you specify this clause, you create an index-organized table .
<i>Italics</i>	Italic typeface indicates book titles or emphasis.	<i>Oracle8i Concepts</i> Ensure that the recovery catalog and target database do <i>not</i> reside on the same disk.
UPPERCASE monospace (fixed-width font)	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a NUMBER column. You can back up the database by using the BACKUP command. Query the TABLE_NAME column in the USER_TABLES data dictionary view. Use the DBMS_STATS.GENERATE_STATS procedure.

Convention	Meaning	Example
lowercase monospace (fixed-width font)	<p>Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values.</p> <p>Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.</p>	Enter <code>sqlplus</code> to open SQL*Plus. The password is specified in the <code>orapwd</code> file. Back up the datafiles and control files in the <code>/disk1/oracle/dbs</code> directory. The <code>department_id</code> , <code>department_name</code> , and <code>location_id</code> columns are in the <code>hr.departments</code> table. Set the <code>QUERY_REWRITE_ENABLED</code> initialization parameter to <code>true</code> . Connect as <code>oe</code> user. The <code>JRepUtil</code> class implements these methods.
lowercase monospace (fixed-width font) italic	Lowercase monospace italic font represents placeholders or variables.	You can specify the <code>parallel_clause</code> . Run <code>Uold_release.SQL</code> where <code>old_release</code> refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
[]	Brackets enclose one or more optional items. Do not enter the brackets.	<code>DECIMAL (digits [, precision])</code>
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	<code>{ENABLE DISABLE}</code>
	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	<code>{ENABLE DISABLE}</code> <code>[COMPRESS NOCOMPRESS]</code>

Convention	Meaning	Example
...	Horizontal ellipsis points indicate either: <ul style="list-style-type: none"> ■ That we have omitted parts of the code that are not directly related to the example ■ That you can repeat a portion of the code 	<pre>CREATE TABLE ... AS <i>subquery</i>;</pre> <pre>SELECT <i>col1</i>, <i>col2</i>, ... , <i>coln</i> FROM employees;</pre>
.	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	<pre>acctbal NUMBER(11,2);</pre> <pre>acct CONSTANT NUMBER(4) := 3;</pre>
<i>Italics</i>	Italicized text indicates placeholders or variables for which you must supply particular values.	<pre>CONNECT SYSTEM/<i>system_password</i></pre> <pre>DB_NAME = <i>database_name</i></pre>
UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	<pre>SELECT last_name, employee_id FROM employees;</pre> <pre>SELECT * FROM USER_TABLES;</pre> <pre>DROP TABLE hr.employees;</pre>
lowercase	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files. Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	<pre>SELECT last_name, employee_id FROM employees;</pre> <pre>sqlplus hr/hr</pre> <pre>CREATE USER mjones IDENTIFIED BY ty3MU9;</pre>

Introduction to Oracle Web Cache

This chapter describes the performance barriers faced by Web sites and introduces the technology which 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 Web Caching Works](#)
- [Benefits of Web Caching](#)
- [Features of Oracle Web Cache](#)

What is the Big Picture for Caching?

The e-business model creates new performance requirements for Web sites. To carry out electronic business successfully, Web sites must protect against poor response time and system outages caused by peak loads. Slow performance translates into lost revenue.

Many high-volume Web sites 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 application Web servers will have to be added. In short, the manageability costs associated with adding application Web 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.

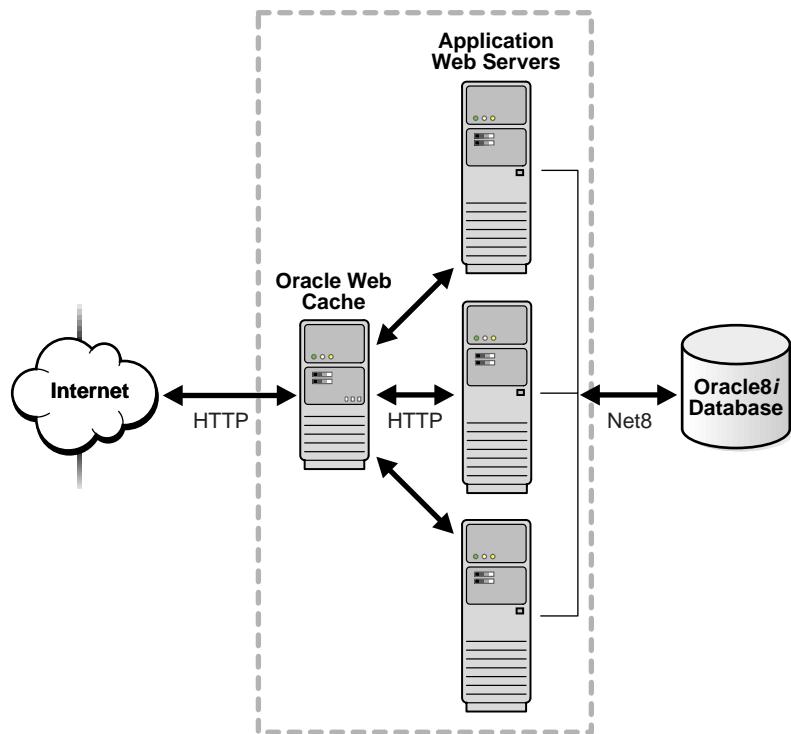
Oracle's Solution to Web Site Performance Issues

Faced with these performance challenges, e-businesses need to invest in more cost-effective technologies and services to improve the performance of their sites. Oracle offers Oracle Web Cache to help e-businesses manage Web site performance issues. Oracle Web Cache is a content-aware server accelerator, or **reverse proxy server**, that improves the performance, scalability, and availability of Web sites that run on Oracle9*i* Application Server and Oracle8*i*.

By storing frequently accessed URLs in virtual memory, Oracle Web Cache eliminates the need to repeatedly process requests for those URLs on the application Web server. Unlike legacy proxy servers that handle only static documents, Oracle Web Cache caches both static and dynamically generated content from one or more application Web servers. Oracle Web Cache retrieves content faster than legacy proxies and greatly reduces the load on application Web servers.

Figure 1–1 shows the basic architecture. Oracle Web Cache sits in front of application Web servers, caching their content, and providing that content to Web browsers that request it. When Web browsers access the Web site, they send HTTP requests to Oracle Web Cache. Oracle Web Cache, in turn, acts as a virtual server to the application Web servers. If the requested content has changed, Oracle Web Cache retrieves the new content from the application Web servers. The application Web servers may retrieve their content from an Oracle database.

Figure 1–1 Oracle Web Cache Architecture



How Web Caching Works

To Web browsers, Oracle Web Cache acts as the virtual server for application Web servers. You configure Oracle Web Cache with the same IP address that is registered for a site's domain name and the application Web servers' host names. This

configuration enables Web browsers to communicate with Oracle Web Cache rather than application Web servers when accessing a Web site.

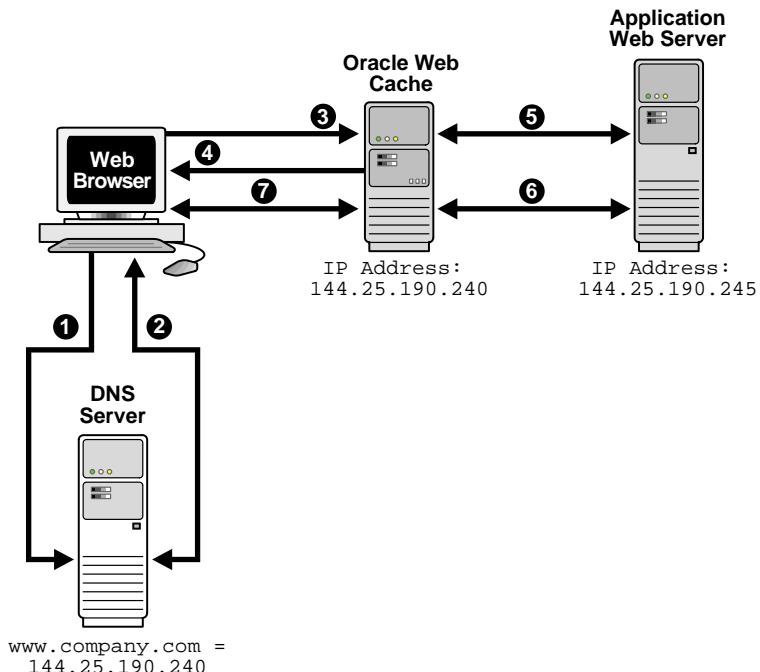
Figure 1–2 on page 1–5 shows how Web caching works. Oracle Web Cache has an IP address of 144.25.190.240 and the application Web server has an IP address of 144.25.190.245. The steps for browser interaction with Oracle Web Cache follow:

1. A browser sends a request to a Web site named `www.company.com`.
This request in turn generates a request to Domain Name System (DNS) for the IP address of for the Web site.
2. DNS returns the IP address of Oracle Web Cache, that is, 144.25.190.240.
3. The browser sends the request for the Web page to Oracle Web Cache, 144.25.190.240.
4. If the requested content is in its cache, Oracle Web Cache sends the content directly to the browser. This is called a **cache hit**.

Note: Dynamic content is generated by the application Web server and then returned to Oracle Web Cache before being passed to the browser.

5. If Oracle Web Cache does not have the requested content or the content is stale or invalid, it hands the request off to the application Web server. This is called a **cache miss**.
6. The application Web server sends the content through Oracle Web Cache.
7. Oracle Web Cache sends the content to the client and makes a copy of the page in cache.

Note: A page stored in the cache is removed when it becomes invalid or outdated, as described in "[Cache Freshness and Performance Assurance](#)" on page 2–2.

Figure 1–2 Web Server Acceleration

Benefits of Web Caching

Web caching provides the following benefits for Web sites:

- Performance

Running on inexpensive hardware, Oracle Web Cache can increase the throughput of a Web site by several orders of magnitude. In addition, Oracle Web Cache significantly reduces response time to browser requests by storing documents in memory and by serving compressed versions of documents to browsers that support the GZIP encoding method.

- Scalability

In addition to unparalleled throughput, Oracle Web Cache can sustain thousands of concurrent browser connections, meaning that visitors to a site see fewer application Web server errors, even during periods of peak load.

- **High Availability**

Oracle Web Cache supports content-aware (Layer 4 - Layer 7) load balancing and failover detection. These features ensure that cache misses are directed to the most available, highest-performing Web server in the cluster. Moreover, a patent-pending capacity heuristic guarantees performance and provides surge protection when 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, Oracle Web Cache offers a simple and inexpensive means of reducing a Web site's cost per request.

- **Network Traffic Reduction**

Most requests are resolved by Oracle Web Cache, reducing traffic to the application Web servers. The cache also reduces traffic to backend databases located on computers other than the application Web server.

Features of Oracle Web Cache

The main features of Oracle Web Cache make it a perfect caching service for e-business Web sites that host online catalogs, news services, and portals. These features include:

- [Static and Dynamically Generated Content Caching](#)
- [Cache Invalidation](#)
- [Performance Assurance](#)
- [Surge Protection of Application Web Servers](#)
- [Load Balancing of Application Web Servers](#)
- [Security Features](#)
- [Administration](#)
- [Compression](#)

Static and Dynamically Generated Content Caching

Oracle Web Cache uses cacheability rules to store documents. These rules fall into two categories:

- Rules for static content, such as GIF, JPEG, or static HTML files
- Rules for dynamically generated content created using technologies like Java Server Pages (JSP), Active Server Pages (ASP), PL/SQL Server Pages (PSP), Java Servlets, and Common Gateway Interface (CGI). Support of these technologies enables Oracle Web Cache to recognize rules for the following:
 - Multiple-version documents 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

See Also: ["Caching Dynamically Generated Content"](#) on page 2-5 for further information about dynamically-generated content

Cache Invalidation

Oracle Web Cache supports **invalidation** as a mechanism to keep its cache consistent with the content on the application Web servers, origin databases, or other dynamically generated means.

Administrators can invalidate cache content in one of two ways:

- Send an HTTP invalidation message to the computer running Oracle Web Cache

When documents are invalidated and a browser requests them, Oracle Web Cache refreshes them with new content from the application Web server.

- Assign an expiration time limit to the documents

When a document expires, Oracle Web Cache treats it like an invalid document, that is, if requested by a browser, it refreshes it with a updated content from the application Web server.

See Also: "Cache Freshness and Performance Assurance" on page 2-2 for further information about invalidation

Performance Assurance

When a large number of documents have been invalidated, the retrieval of a new documents can result in overburdened application Web servers.

To handle performance issues while maintaining cache consistency, Oracle Web Cache uses built-in **performance assurance heuristics** that enable it to assign a queue order to documents. These heuristics determine which documents can be served stale and which documents must be refreshed immediately. Documents with a higher priority are refreshed first. Documents with a lower priority are refreshed at a later time.

The queue order of documents is based on the popularity of documents and the validity of documents assigned during invalidation. If the current load and capacity of the application Web server is not exceeded, the most popular and least valid documents are refreshed first.

See Also: "Cache Freshness and Performance Assurance" on page 2-2 for further information about performance assurance

Surge Protection of Application Web Servers

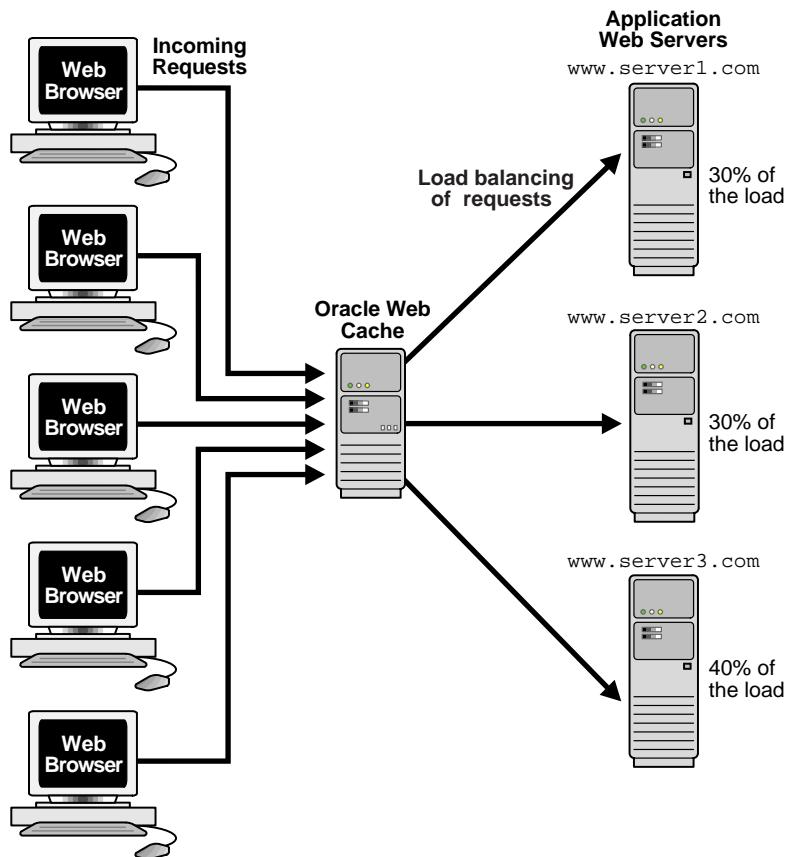
Oracle Web Cache passes requests for non-cacheable or stale documents to the application Web servers. To prevent an overload of requests on the application Web servers, Oracle Web Cache has a surge protection feature that enables you to set a limit on the number of concurrent requests that the application Web servers can handle. When the limit is reached, subsequent requests are queued to wait up to a maximum amount of time. If the maximum wait time is exceeded, Oracle Web Cache rejects the request and serves a site busy apology page to the Web browser that initiated the request.

Load Balancing of Application Web Servers

Most Web sites are served by multiple application Web servers running on multiple computers that share the load of HTTP requests. This feature enables Web sites to be built with a collection of servers for better scalability and reliability. Oracle Web Cache is designed to manage HTTP requests for up to 100 application Web servers. All requests that Oracle Web Cache cannot serve are passed to the application Web servers. Oracle Web Cache has a **load balancing** feature that distributes these requests over a set of application Web servers. To configure load balancing, you set the capacity (concurrent connections) for each application Web server. Based on the capacity assigned, Oracle Web Cache prescribes the relative percentage load of each application Web server.

Figure 1–3 shows three application Web servers, whereby 30 percent of traffic goes to `www.server1.com`, 30 percent goes to `www.server2.com`, and 40 percent goes to `www.server3.com`.

Figure 1–3 Load Balancing



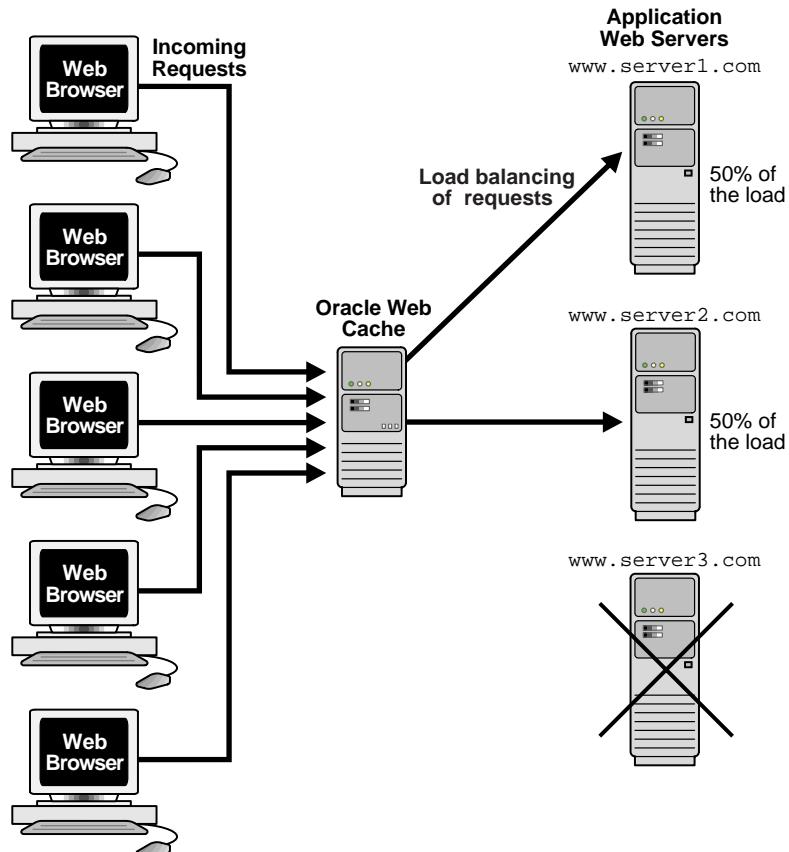
Backend Failover

After five continuous request failures, Oracle Web Cache considers an application Web server as failed. When an application Web server fails, Oracle Web Cache automatically distributes the load over the remaining application Web servers based on the remaining proportion. Oracle Web Cache polls the failed application Web server for its current up/down status every 60 seconds until it is back online.

The **failover** feature is shown in [Figure 1-4](#). An outage of `www.server3.com`, which was receiving 40 percent of requests, results in 50 percent of the requests going to `www.server1.com` and 50 percent of requests going to `www.server2.com`. This is based on a 30 percent load for both `www.server1.com` and `www.server2.com`. If `www.server1.com` were to fail instead, based on a 30 percent load for `www.server2.com` and a 40 percent load for `www.server3.com`, 42.86 percent of the requests would go to `www.server2.com` and 57.14 percent of requests would go to `www.server3.com`.

When the failed server returns to operation, Oracle Web Cache will include it in the load mix as previously prescribed.

Figure 1-4 Failover



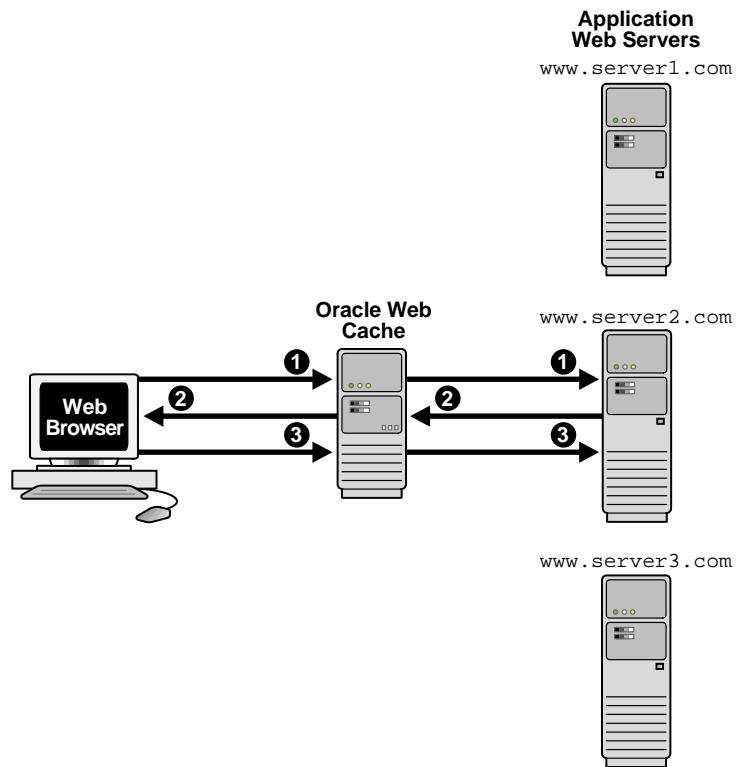
Application Web Server Binding

Oracle Web Cache supports Web sites that use session IDs and/or **session cookies** to bind user sessions to a given application Web server in order to maintain state for a period of time. To utilize the **session binding** feature, the application Web server itself must maintain state, that is, it must be stateful. Web sites bind user sessions by including session data in the HTTP header or body it sends to Web browsers in such a way that the browser is forced to include it with its next request. This data is transferred either with parameters embedded in the URL or cookies, which are text strings stored on the client.

Figure 1–5 shows how Oracle Web Cache supports documents that use application Web server binding:

1. When a request first comes in, Oracle Web Cache uses load balancing to decide which application Web server to send it to. In this example, `www.server2.com` was chosen.
2. If the requested document requires application Web server binding, the application Web server sends the session information back to the browser through Oracle Web Cache in the form of a cookie or an embedded URL parameter.
3. Oracle Web Cache sends subsequent requests for the session to the application Web server that established the session, bypassing load balancing. In this example, `www.server2.com` handles the subsequent requests.

Figure 1–5 Application Web Server Binding



Security Features

Oracle Web Cache provides important security features including:

- Password authentication for administration and invalidation operations
- Control over which ports administration and invalidation operations can be requested from
- Timeout for inactive connections
- IP and subnet administration restrictions

Administration

Oracle Web Cache provides a graphical user interface tool called **Oracle Web Cache Manager** that combines configuration and monitoring options to provide an integrated environment for configuring and managing Oracle Web Cache and the Web sites it caches for. With the Oracle Web Cache Manager, you can easily:

- Configure Oracle Web Cache to cache for application Web servers
- Start and stop Oracle Web Cache
- Establish cacheability rules
- Monitor Oracle Web Cache and Web site performance
- Establish listening ports and security passwords

Compression

You can select to have documents compressed upon insertion into the cache. Oracle Web Cache is able to compress 300 KB files down to 3 KB, increasing the overall throughput by a factor of 100.

2

Oracle Web Cache Concepts

This chapter explains how Oracle Web Cache is populated with content, how that content maintains consistency, and how dynamically generated content is cached. This chapter contains these topics:

- [Populating Oracle Web Cache](#)
- [Cache Freshness and Performance Assurance](#)
- [Caching Dynamically Generated Content](#)

Populating Oracle Web Cache

Oracle Web Cache uses cacheability rules to determine which documents to cache. When Oracle Web Cache is first configured with a cacheability rule for a set of documents, those documents are not initially in the cache until there is a browser request for those documents. When a request comes in, Oracle Web Cache sends the request to the application Web server. If the requested document is specified as one of the documents to cache, Oracle Web Cache caches the document for subsequent requests.

If a document contains a **cookie**, Oracle Web Cache evaluates the cookie value of the browser request and application Web server response. If the values match and there is a corresponding cacheability rule, Oracle Web Cache caches the response. However, **session cookie** values are not evaluated. For documents that use these cookies, the response is cached, regardless if the cookie values match.

Note: The cache can also be populated with Apache Benchmark tool. See your Apache documentation for further information.

See Also: ["Caching Dynamically Generated Content" on page 2-5](#) for an overview of cookies

Cache Freshness and Performance Assurance

Consistency and performance are crucial for the reliability of Oracle Web Cache.

Invalidation and **expiration** ensure consistency between the cache and the content on the application Web servers. With invalidation, a HTTP message is sent by specifying which documents to mark as invalid. With expiration, documents are marked as invalid after a certain amount of time in the cache. When documents are marked as invalid and a browser requests them, they are either immediately removed and refreshed or refreshed whenever the application Web servers can refresh them.

Expirations are useful if it can be accurately predicated when content will change on an application Web server or database. An invalidation messages is intended for less predictable, more frequently changing content.

One could logically assume that widespread cache invalidation or expiration would negatively impact performance of the application Web servers, resulting in the generation of `HTTP 503 Server Busy` errors in the access log file or even application Web server overload. For this reason, Oracle Web Cache intelligently serves some of the documents stale until the application Web servers have the capacity to refresh them.

Oracle Web Cache provides minimal trade-off between performance and consistency through **performance assurance heuristics** that determine which documents can be served stale. These heuristics are based on a number of factors including:

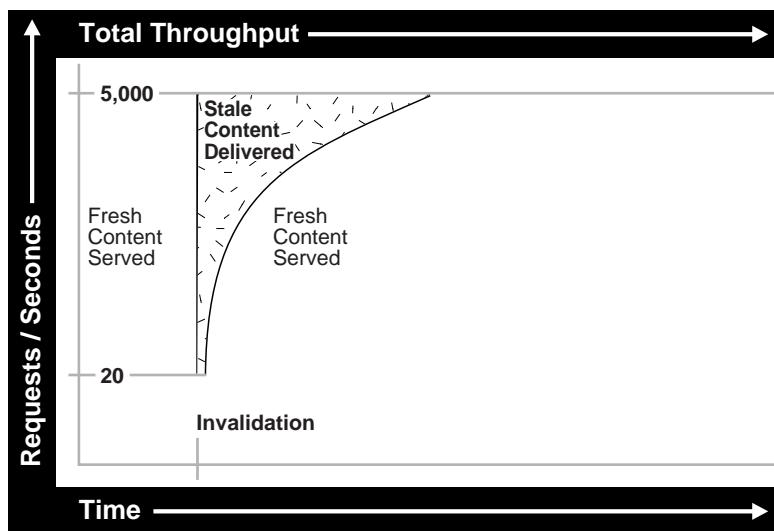
Popularity of Document	Popularity is determined by: <ul style="list-style-type: none">■ The number of times the documentation has been requested since being placed in the cache■ The number of recent requests for the document
Validity of Document	The level assigned during invalidation or expiration. The higher the validity level, the longer Oracle Web Cache serves these documents stale from the cache before invalidating them. For documents with lower validity levels, Oracle Web Cache serves these documents stale for a short amount of time before invalidating them.
Load of the Application Web Server	Critical documents should be assigned a low validity level, and non-critical documents should be assigned a higher validity level.
Limit on the Application Web Server	The current load on the application Web server is determined by the number of open connections from Oracle Web Cache to the application Web servers, that is, the total number of pending requests to the application Web servers.
	The configured limit on the application Web server load is the configured number of concurrent connections the application Web server can have.

Together, these factors provide Oracle Web Cache with a logical queue of content to update from the application Web servers.

Note: Performance assurance heuristics apply when you configure documents to be refreshed based on when the application Web servers can refresh them; performance assurance heuristics do not apply when documents are immediately removed.

Figure 2–1 illustrates how performance assurance heuristics are used during widespread invalidation. At the beginning of the invalidation curve, the number of fresh cache hits decrease to 20 and the number of stale cache hits increase to 4,980. However, during invalidation, the number of fresh cache hits quickly increase. This is because Oracle Web Cache refreshes the most popular documents first so that these documents have little chance of being served stale. Once the popular documents are refreshed, the less popular documents are refreshed. The total number of documents that can revalidated in a given period of time is dependent on application Web server capacity. At the end of invalidation, only fresh content is served.

Figure 2–1 Performance Assurance Heuristics Graph



Caching Dynamically Generated Content

Because of invalidation, Oracle Web Cache knows what documents are valid and what documents are invalid. This is especially important for dynamically generated content that changes frequently. Dynamically generated content is created using technologies like Java Server Pages (JSP), Active Server Pages (ASP), PL/SQL Server Pages (PSP), Java Servlets, and Common Gateway Interface (CGI). Examples of pages that are dynamically generated include:

- A Web site's product catalog, where information on pricing and inventory might vary from one moment to the next
- Auction views, which must be regenerated after each successful bid is processed
- Search results, which can change as catalog items are added and removed

Most static caches and content distribution services have no mechanism to verify the consistency of dynamically generated Web pages with the data sources used to create them. Therefore, it is difficult for these services to know when content has changed. Oracle Web Cache, on the other hand, receives invalidation messages from the application Web server, containing the original content.

For dynamically generated pages, browsers pass information about themselves to the application Web server, enabling the application Web server to serve appropriate content to the browser.

The HTTP protocol has a way for browsers and application Web servers to share information, such as session or category information, in message headers that browsers pass with every request to the application Web server. This message header can contain a **cookie**.

Cookies are stored on the browser's file system and are often used for identifying users who revisit Web sites. Many users choose to disable cookies in their browsers out of privacy concerns. For this reason, application Web servers often embed parameter information in the URL.

Oracle Web Cache is able to recognize both cookies and embedded URL parameters, enabling it to recognize cacheability rules for pages with:

- [Multiple Versions of the Same URL](#)
- [Personalized Attributes](#)
- [Session Information](#)

See Also: <http://www.cookiecentral.com/> for further information about cookies

Multiple Versions of the Same URL

Some pages have multiple versions of the same URL, enabling categorization.

Figure 2-2 shows the same URL,

http://store.oracle.com/cec/cstage?eccookie=&ecsid=1225&ecaction=ecproditemlistbysupersect&template=decsectview_mp_en.htm, with different prices for customers and internal Oracle employees. While customers pass a cookie name and value of `ec-400-id-acctcat=WALKIN`, employees pass a cookie name and value of `ec-400-id-acctcat=CUSTOMER`.

Figure 2-2 Multiple-Version URL

The diagram illustrates two versions of the Oracle Store website, each with a user icon and a 'Unit Price' label pointing to the price column in the product table.

Oracle Customer Version:

Product	Unit Price	# of Units	Buy
Oracle Internet Developer Suite Rel 1.0 CD Pack for MS Windows NT	\$39.95	<input type="checkbox"/>	Buy
Oracle Database Bi Release 2 (8.1.6) CD Pack v5 for AIX-Based Systems	\$39.95	<input type="checkbox"/>	Buy
Oracle Database Bi Release 2 (8.1.6) CD Pack v5 for Compaq Tru64 UNIX	\$39.95	<input type="checkbox"/>	Buy
Oracle Database Bi Release 2 (8.1.6) CD Pack v5 for HP 9000 Series HP-UX	\$39.95	<input type="checkbox"/>	Buy
Oracle Database Bi Release 2 (8.1.6) CD Pack v6 for Linux Intel	\$39.95	<input type="checkbox"/>	Buy
Oracle Database Bi Release 2 (8.1.6) CD Pack v4 for MS Windows NT	\$39.95	<input type="checkbox"/>	Buy
Oracle Database Bi Release 2 (8.1.6) CD Pack v5 for Sun Solaris Intel	\$39.95	<input type="checkbox"/>	Buy
Oracle Database Bi Release 2 (8.1.6) CD Pack v7 for Sun SPARC Solaris	\$39.95	<input type="checkbox"/>	Buy
Oracle Database Bi (8.1.5) CD Pack v7 for AIX-Based Systems	\$39.95	<input type="checkbox"/>	Buy
Oracle Database Bi (8.1.5) CD Pack v7 for Compaq Tru64 UNIX	\$39.95	<input type="checkbox"/>	Buy
Oracle Database Bi (8.1.5) CD Pack v6 for HP 9000 Series HP-UX	\$39.95	<input type="checkbox"/>	Buy
Oracle Database Bi (8.1.5) CD Pack v5 for Linux Intel	\$39.95	<input type="checkbox"/>	Buy
Oracle Database Bi (8.1.5) CD Pack v9 for MS Windows NT	\$39.95	<input type="checkbox"/>	Buy

Oracle Employee Version:

Product	Unit Price	# of Units	Buy
Oracle Internet Developer Suite Rel 1.0 CD Pack for MS Windows NT	\$0.00	<input type="checkbox"/>	Buy
Oracle Database Bi Release 2 (8.1.6) CD Pack v5 for AIX-Based Systems	\$0.00	<input type="checkbox"/>	Buy
Oracle Database Bi Release 2 (8.1.6) CD Pack v5 for Compaq Tru64 UNIX	\$0.00	<input type="checkbox"/>	Buy
Oracle Database Bi Release 2 (8.1.6) CD Pack v5 for HP 9000 Series HP-UX	\$0.00	<input type="checkbox"/>	Buy
Oracle Database Bi Release 2 (8.1.6) CD Pack v6 for Linux Intel	\$0.00	<input type="checkbox"/>	Buy
Oracle Database Bi Release 2 (8.1.6) CD Pack v4 for MS Windows NT	\$0.00	<input type="checkbox"/>	Buy
Oracle Database Bi Release 2 (8.1.6) CD Pack v5 for Sun Solaris Intel	\$0.00	<input type="checkbox"/>	Buy
Oracle Database Bi Release 2 (8.1.6) CD Pack v7 for Sun SPARC Solaris	\$0.00	<input type="checkbox"/>	Buy
Oracle Database Bi (8.1.5) CD Pack v7 for AIX-Based Systems	\$0.00	<input type="checkbox"/>	Buy
Oracle Database Bi (8.1.5) CD Pack v7 for Compaq Tru64 UNIX	\$0.00	<input type="checkbox"/>	Buy
Oracle Database Bi (8.1.5) CD Pack v6 for HP 9000 Series HP-UX	\$0.00	<input type="checkbox"/>	Buy
Oracle Database Bi (8.1.5) CD Pack v5 for Linux Intel	\$0.00	<input type="checkbox"/>	Buy
Oracle Database Bi (8.1.5) CD Pack v9 for MS Windows NT	\$0.00	<input type="checkbox"/>	Buy

You can configure Oracle Web Cache to recognize and cache multiple-version pages by using the:

- Values of the cookie for the page
- **HTTP request headers** for the page

For those URLs that use a cookie (sometimes referred to as a **category cookie**), you set cacheability rules that specify the cookie name and whether to cache versions of the URL that do not use the cookie.

When a browser sends a request to an application Web server for a multiple-version document and the value of the browser's cookie matches the value of the application Web server's response, the version of the document is cached. If the cookie values do not match, then the version of the document is not cached. Once versions of the document are cached, Oracle Web Cache uses the value of the cookie in the browser's request to serve the appropriate version of the URL to the browser.

Table 2–1 shows four different versions of same URL, `http://www.dot.com/page1.htm`. The URL uses a cookie named `user_type`, which supports browser requests that contain cookie values of `Customer`, `Internal`, and `Promotional`. You can configure Oracle Web Cache to recognize the `user_type` cookie, enabling Oracle Web Cache to cache three different documents. In addition, you can configure Oracle Web Cache to cache a fourth document for those requests that do not use a cookie.

Table 2–1 Multiple-Version URL with Different Cookie Values

Version	URL	Cookie Name/Value
1	<code>http://www.dot.com/page1.htm</code>	<code>user_type=Customer</code>
2	<code>http://www.dot.com/page1.htm</code>	<code>user_type=Internal</code>
3	<code>http://www.dot.com/page1.htm</code>	<code>user_type=Promotional</code>
4	<code>http://www.dot.com/page1.htm</code>	No cookie

For those URLs that use HTTP request headers, you set cacheability rules that specify the HTTP request header whose values to use for disambiguation. HTTP request headers enable Web browsers to pass additional information about the request and about themselves. Oracle Web Cache uses the header to serve the appropriate version of the URL to browsers.

Table 2–2 lists the standard HTTP request headers supported.

Table 2–2 HTTP Request Headers

Header	Description
Accept	Specifies which media types are acceptable for the response Example: Accept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Charset	Specifies which character sets are acceptable for the response Example: Accept-Charset: iso-8859-1,* ,utf-8
Accept-Encoding	Restricts the content-encodings that are acceptable in the response Example: Accept-Encoding: gzip
Accept-Language	Specifies the set of languages that are preferred as a response Example: Accept-Language: en
User-Agent	Contains information about the Web browser that initiated the request Example: User-Agent: Mozilla/4.61 [en] (WinNT; U)

Note: Oracle Web Cache does not interpret the values of these HTTP request headers. If the values for two pages are different, Oracle Web Cache caches both pages separately. For example, if one request sends a HTTP request header of User-Agent:
Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0) and another request sends a HTTP request header of User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT; DigExt) for a different versions of Internet Explorer, Oracle Web Cache serves two pages for the two requests.

Personalized Attributes

Many Web sites support pages with personalized attributes, such as personalized greetings in the form of "Welcome <your name>", icons, addresses, or shopping cart snippets, on an otherwise generic page. You can configure the page with the personalized attributed information contained within HTML tags <!-- WEBCACHETAG> and <!-- WEB CACHEEND--> that Oracle Web Cache can parse. The personalized attribute information can be included within any valid HTML segment that has beginning (<tag>) and ending (</tag>) HTML tags.

Oracle Web Cache reads these tags and caches the instructions for substituting values for personalized attributes based on the information contained within a cookie or an embedded URL parameter.

This functionality enables Oracle Web Cache to use the same page for multiple users. Because only one page needs to be cached, only one application Web server request is required to initially populate the cache with the page. The initial request sets the personalized attribute cookie or embedded URL parameter. All subsequent requests for the page that pass the cookie or embedded URL parameter are served from the cache.

Figure 2-3 on page 2-10 shows two users, Jane Doe and John Doe, accessing the same page,

`http://store.oracle.com/cec/cstage?eccookie=&ecaction=ecpassthru2&template=walkin1.en.htm`. This page contains a personalized greeting suited for the user. The HTML code for the personalized greeting **Jane Doe** uses the following HTML code:

```
<b>
<!-- WEBCACHETAG="person01" -->
Jane Doe
<!-- WEB CACHEEND-->
</b>
```

The HTML code for personalized greeting **John Doe** uses the following HTML code:

```
<b>
<!-- WEBCACHETAG="person01" -->
John Doe
<!-- WEB CACHEEND-->
</b>
```

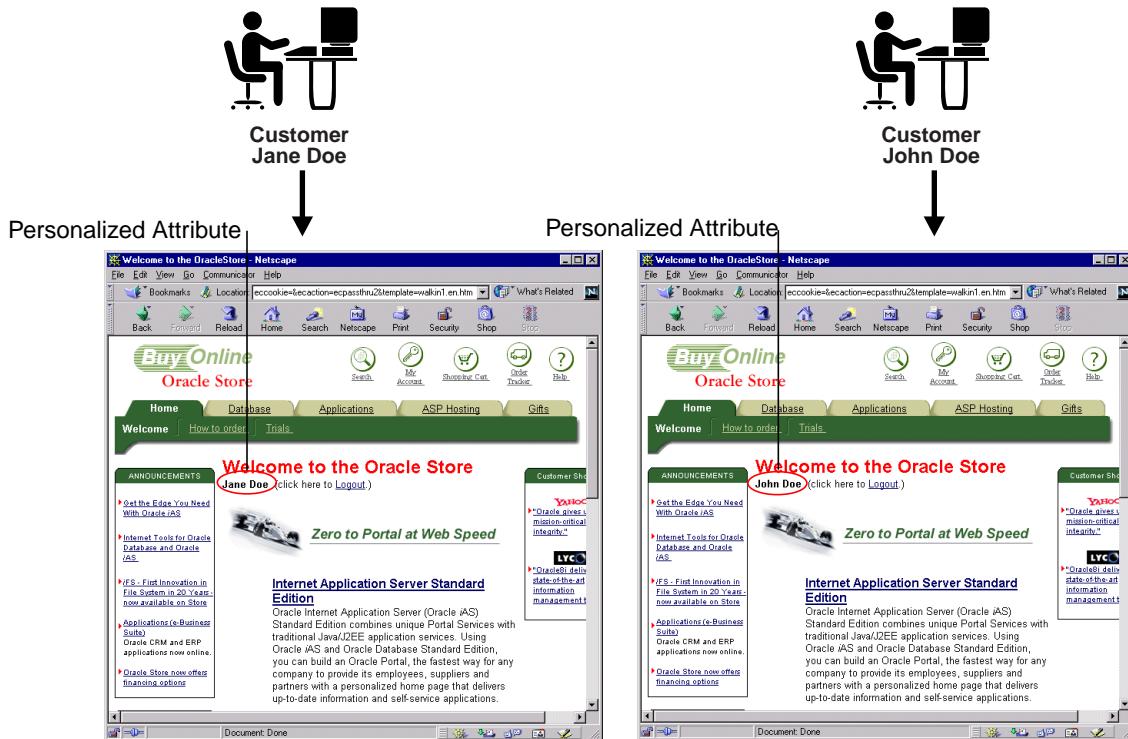
`person01` represents the session name assigned to the `person_name` cookie that Jane and John pass to Oracle Web Cache. Jane passes a cookie name value pair of `person_name=Jane Doe` and John Doe passes a cookie name value pair of `person_name=John Doe`. When Oracle Web Cache receives the cookie

information from Jane and John, it maps the `person_name` cookie to the `person01` session name and substitutes the cookie value.

If, instead of cookies, the page supported embedded URL parameters, then the URL would contain the `person_name` parameter. For example, the page for Jane Doe could be `http://store.oracle.com/cec/cstage?person_name=Jane+Doe` and the page for John Doe could be

`http://store.oracle.com/cec/cstage?person_name=John+Doe`. Oracle Web Cache is configured with the `person_name01` session, which maps to the `person_name` embedded URL parameter. Oracle Web Cache uses the value of the embedded parameter to substitute the appropriate name.

Figure 2–3 Page with a Personalized Attribute



If a request does not contain the cookie or embedded URL parameter, Oracle Web Cache substitutes the personalized attribute with an empty string. If you want to instead always require value of the personalized attribute, then set a session-related caching rule and require that the request get the cookie or embedded URL parameter settings from the application Web server.

See Also: "Configuring Rules for Personalized Pages" on page 6-16

Session Information

Some Web sites keep track of user sessions by assigning each user a unique session ID. Session IDs are typically used for Web sites with catalog pages. The session ID can be used for either **session tracking** or **session-encoded URLs**.

When a user first accesses a Web site that uses session IDs, Oracle Web Cache passes the request to the application Web server to establish the session. In turn, the application Web server assigns the user with a session ID through a cookie (sometimes referred to as a **session cookie**) or an embedded in the URL as a parameter. As users request pages and HTML links that use these session cookies or embedded URL parameters, application Web server track the sessions. You can configure Oracle Web Cache to serve pages that support session tracking and session-encoded URLs.

Session Tracking

Because session tracking does not alter the actual content of a page, you can configure Oracle Web Cache to cache the page and serve it to multiple users.

If you configure Oracle Web Cache to cache a page that uses session information and a subsequent request for the page contains a session cookie or embedded URL parameter, then Oracle Web Cache serves the page with the user's session information from its cache.

To better understand how session tracking works, consider the HTML pages shown in [Figure 2-3](#) on page 2-10. When Jane Doe and John first access the Oracle Store Web site, their initial requests are sent to the application Web server, which assigns them cookie name value pairs of `session_ID=33436` and `session_ID=33437`, respectively. If their browsers did not support cookies, then the URL for the pages could contain the session ID. For example, the page for Jane Doe would be `http://store.oracle.com/cec/cstage?session_ID=33436` and the page for John Doe could be `http://store.oracle.com/cec/cstage?session_ID=33437`. Oracle Web Cache can be configured to cache one version of this page and other session tracking pages and serve it to multiple users. By using the value of the `session_ID` cookie or embedded URL parameter, Oracle Web Cache can serve the same page to both Jane Doe and John Doe.

Unlike category cookies used for multiple versions of the same URL, Oracle Web Cache ignores the values of session cookies. The response from the application Web server is cached, even if the response session cookie value does not match the request session cookie value. If you do not want the response cached when there is a value mismatch, then modify the application to instead send a non-200 status code as the response.

See Also: "Configuring Rules for Pages with Session Tracking" on page 6-24

Session-Encoded URLs

You can configure Oracle Web Cache to cache the instructions for substituting session information for one user with another based on the session information contained within a cookie or an embedded URL parameter.

Continuing with the example in "Session Tracking" on page 2-11, assume that Jane Doe and John Doe are again assigned cookies or embedded URL parameters of `session_ID=33436` and `session_ID=33437` by the application Web server. The page shown in Figure 2-4 has several `` links that include the `session_ID` cookie or parameter. The **Release 3 (8.1.7)** under the **Oracle8i Documentation** heading for Jane Doe uses the following HTML code:

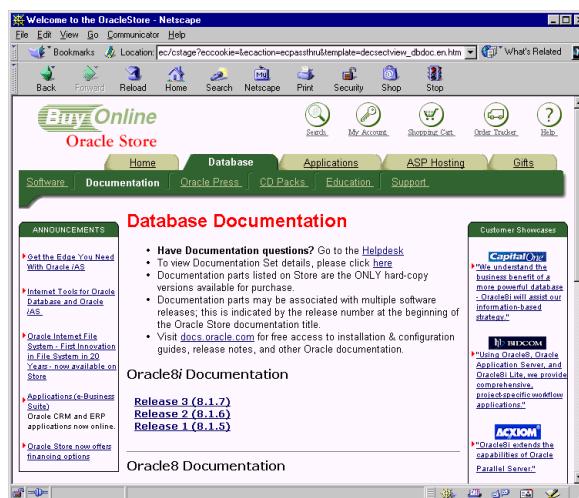
```
<A HREF="/cec/cstage?ecaction=ecproditemlistbysupersect&
ecsid=20330&eccookie=&template=decsectview_pub.en.htm&session_
ID=334326">Release 3 (8.1.7)</A>
```

The same link for John Doe uses the following HTML code:

```
<A HREF="/cec/cstage?ecaction=ecproditemlistbysupersect&
ecsid=20330&eccookie=&template=decsectview_pub.en.htm&session_
ID=334327">Release 3 (8.1.7)</A>
```

By using the value of the `session_ID` cookie or embedded URL parameter, Oracle Web Cache is able to substitute the correct session information for Jane Doe and John Doe.

Figure 2-4 Session-Encoded URLs



Whereas a session tracking page requires that each user establish a session with the application Web server, a page with session-encoded URLs requires that only the initial request to establish a session. Once the cache is populated with the page, other requests are served from the cache, regardless if the request has a session cookie or embedded URL parameter. This has twofold effect for those requests without the session cookie or embedded URL parameter:

- Oracle Web Cache substitutes the session information in the links with an empty string
- Session establishment for the user by the application Web server is delayed until there is a cache miss

If you want to instead require session establishment, then set a session-related caching rule.

See Also: "Configuring Rules for Personalized Pages" on
page 6-16

Deploying Oracle Web Cache

Note: Oracle Web Cache is compatible with Oracle HTTP Server or any other HTTP-compliant application Web server.

This chapter presents several scenarios for deploying Oracle Web Cache. This chapter contains these topics:

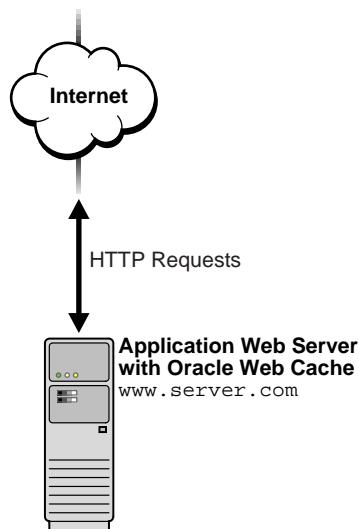
- [Caching Content for One Application Web Server](#)
- [Forwarding HTTPS Requests](#)
- [Load Balancing Requests Among Application Web Servers](#)
- [Accelerating Portions of a Web Site](#)
- [Using Oracle Web Cache Servers in a Failover Pair](#)
- [Working with Firewalls](#)
- [Deploying Oracle Web Cache Servers in a Distributed Network](#)

Caching Content for One Application Web Server

Oracle Web Cache can be deployed on the same computer as the application Web server or on a separate computer.

[Figure 3–1](#) shows Oracle Web Cache deployed on the same computer as the application Web server.

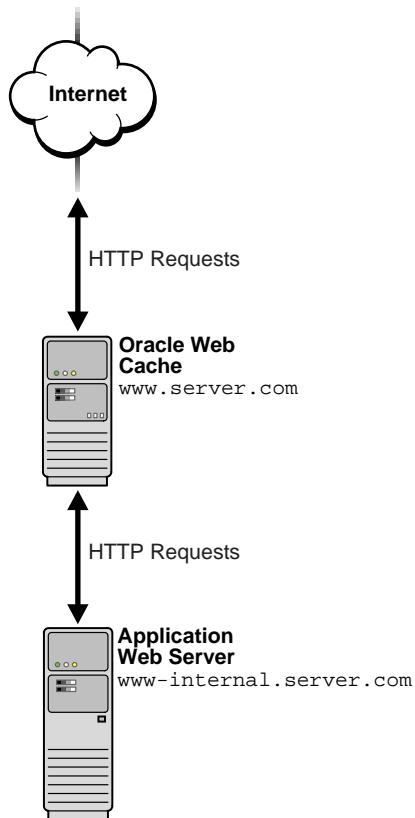
Figure 3–1 Oracle Web Cache On Same Computer As the Application Web Server



For this deployment, configure Oracle Web Cache with the host name of the application Web server.

Figure 3–2 shows Oracle Web Cache deployed on a different computer from the application Web server.

Figure 3–2 Oracle Web Cache On a Different Computer From the Application Web Server



To configure this deployment:

1. Register the IP address of Oracle Web Cache server with the Web site's domain name.
2. Rename the application Web server, and assign the computer running Oracle Web Cache with the name that was previously assigned to the application Web server. In **Figure 3–2**, Oracle Web Cache is named `www.server.com`, which was the name of the application Web server. The application Web server is renamed to `www-internal.server.com`.
3. Configure Oracle Web Cache with the host name of the application Web server.

Forwarding HTTPS Requests

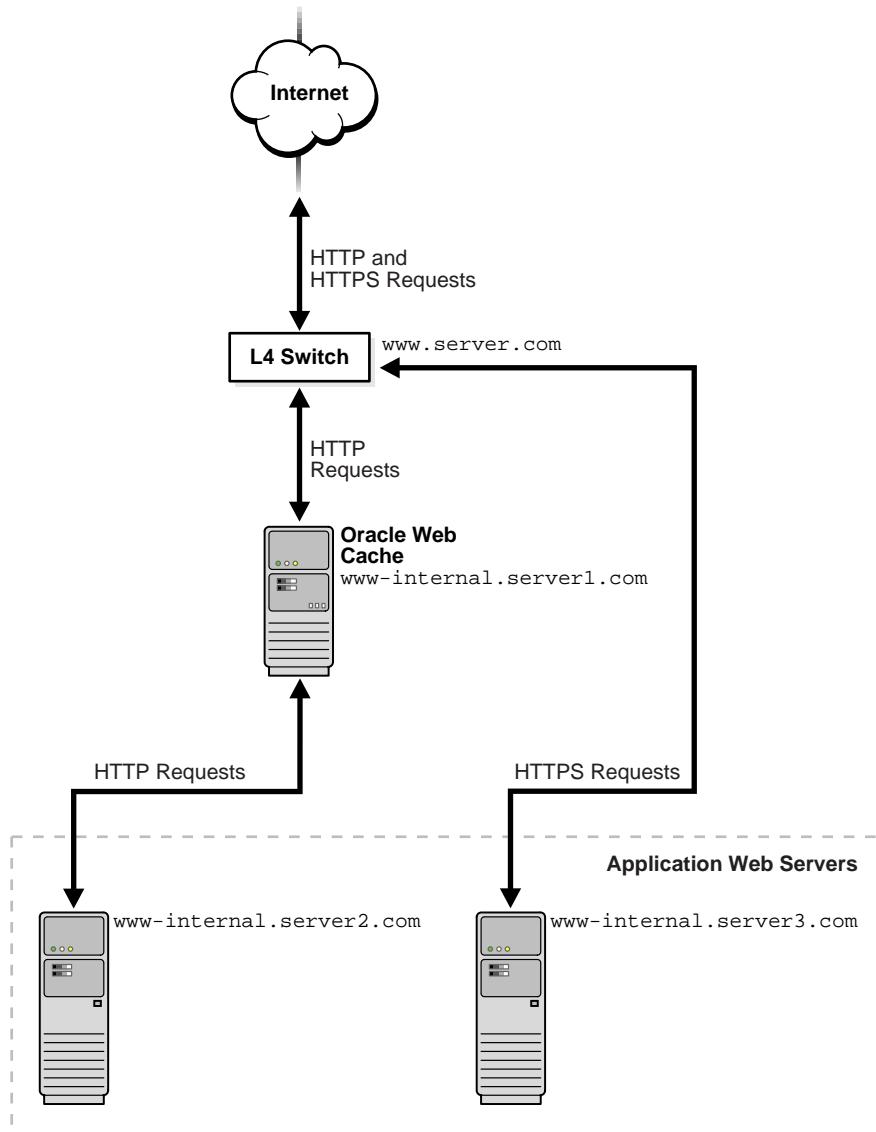
Note: In future releases, you can configure Oracle Web Cache to process both HTTP and HTTPS requests.

In this release, Oracle Web Cache can cache documents for **HTTP protocol** requests. For **HTTPS protocol** requests, you can use a **Layer 4 (L4) switch** to pass all HTTP requests, typically port 80 traffic, to Oracle Web Cache, and forward all HTTPS requests for secure pages to a particular application Web server. [Figure 3–3](#) on page 3-5 shows an L4 switch passing HTTP requests to Oracle Web Cache server `www-internal.server1.com` and HTTPS requests to application Web server `www-internal.server3.com`. Please note that HTTPS requests could also be passed to `www-internal.server2.com`.

An L4 switch operates at Layer 4, the Transport (or protocol) layer, of the **Open Systems Interconnection (OSI)** model. L4 switches determine where to send requests based on the protocol and port number.

See Also: <http://www.ietf.org/> for information about the OSI stack

Figure 3–3 Using an L4 Switch to Forward HTTPS Requests



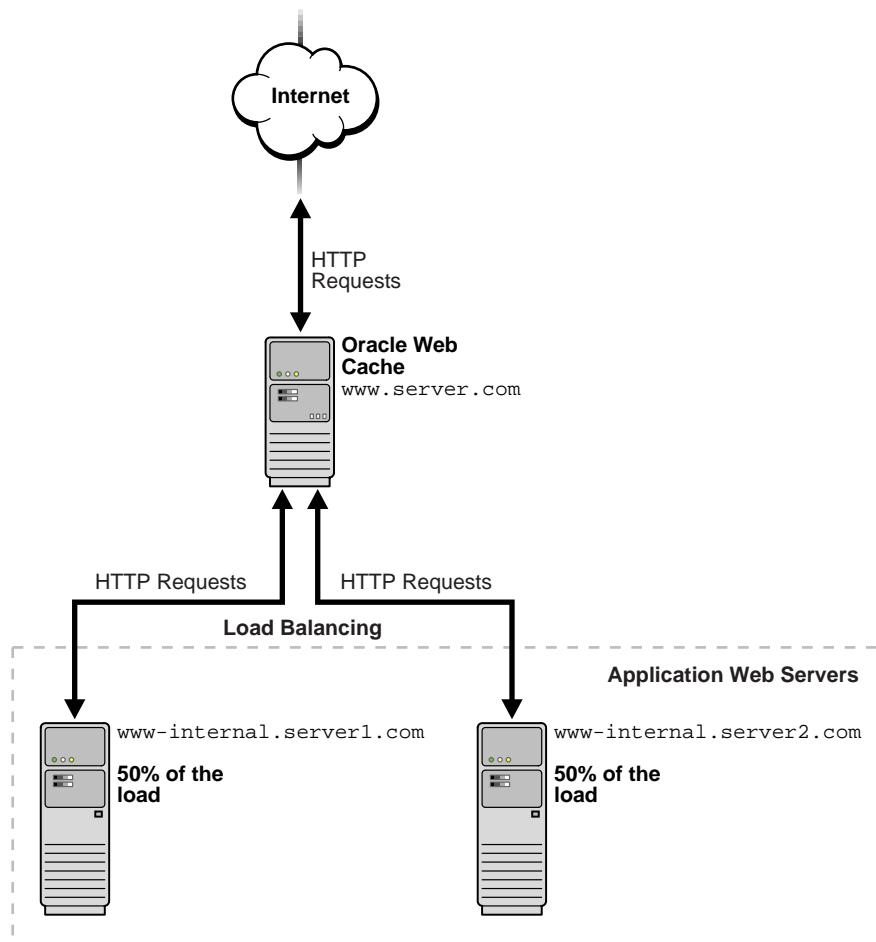
To configure this deployment:

1. Configure the L4 switch with the host names of Oracle Web Cache server for HTTP requests and the application Web server for HTTPS requests.
2. Configure Oracle Web Cache with the host names of the application Web servers.

Load Balancing Requests Among Application Web Servers

Many of today's Web sites use a Load Balancer to balance the incoming requests among multiple Web servers. Instead, as shown in [Figure 3–4](#), you can use Oracle Web Cache to distribute HTTP requests among two or more application Web servers.

Figure 3–4 Load Balancing with Oracle Web Cache

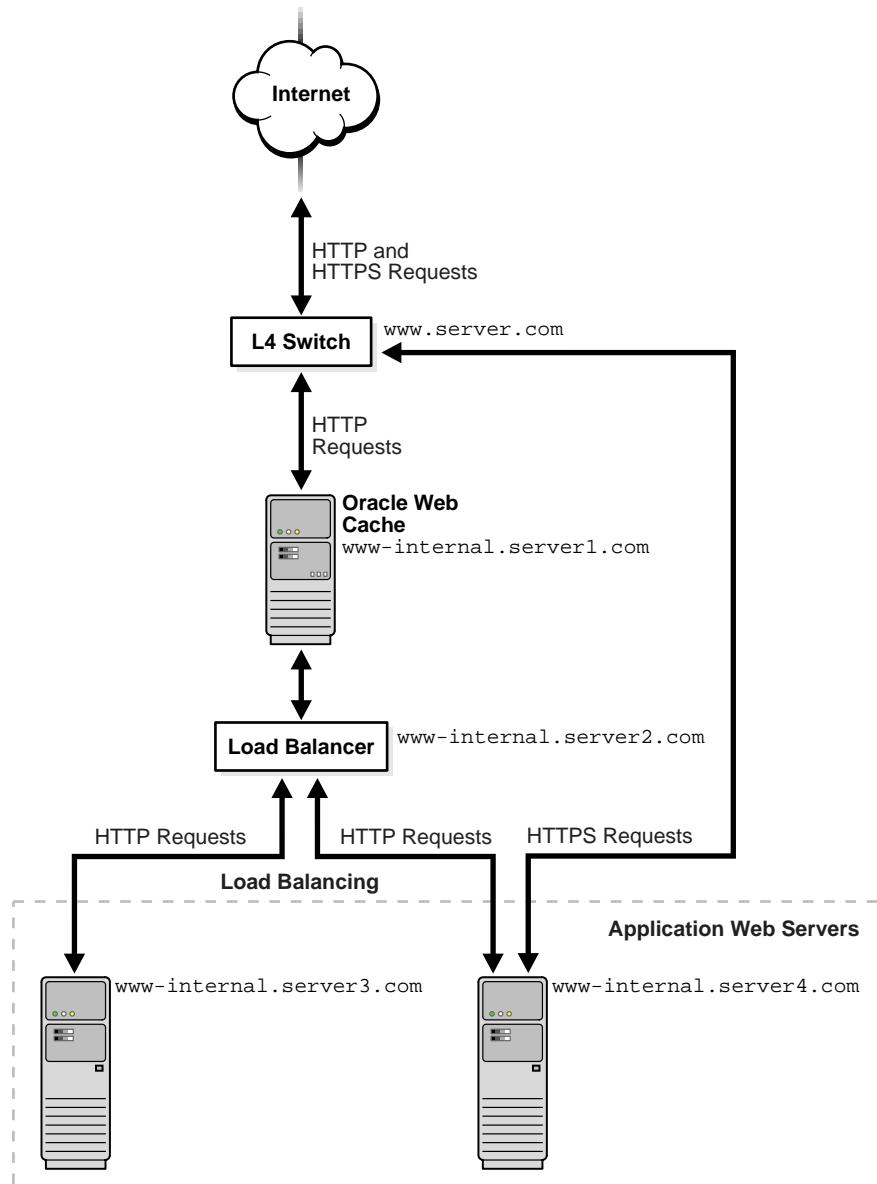


To configure this deployment:

1. Assign the name of the Load Balancer to Oracle Web Cache.
2. Configure Oracle Web Cache with the host names of the application Web servers.

If you prefer to use a third-party Load Balancer, you can use the deployment depicted in [Figure 3–5](#). In this deployment, an L4 switch sends HTTP requests to Oracle Web Cache server `www-internal.server1.com` and HTTPS requests to application Web server `www-internal.server4.com`. The Load Balancer distributes requests from Oracle Web Cache among the pool of application Web servers.

Figure 3–5 Load Balancing with a Third-Party Load Balancer



To configure this deployment:

1. Configure the L4 switch with the host name of Oracle Web Cache server for HTTP requests and the application Web server for HTTPS requests.
2. Configure Oracle Web Cache with the host name of the Load Balancer.
3. Configure the second Load Balancer with the host names of the application Web servers.

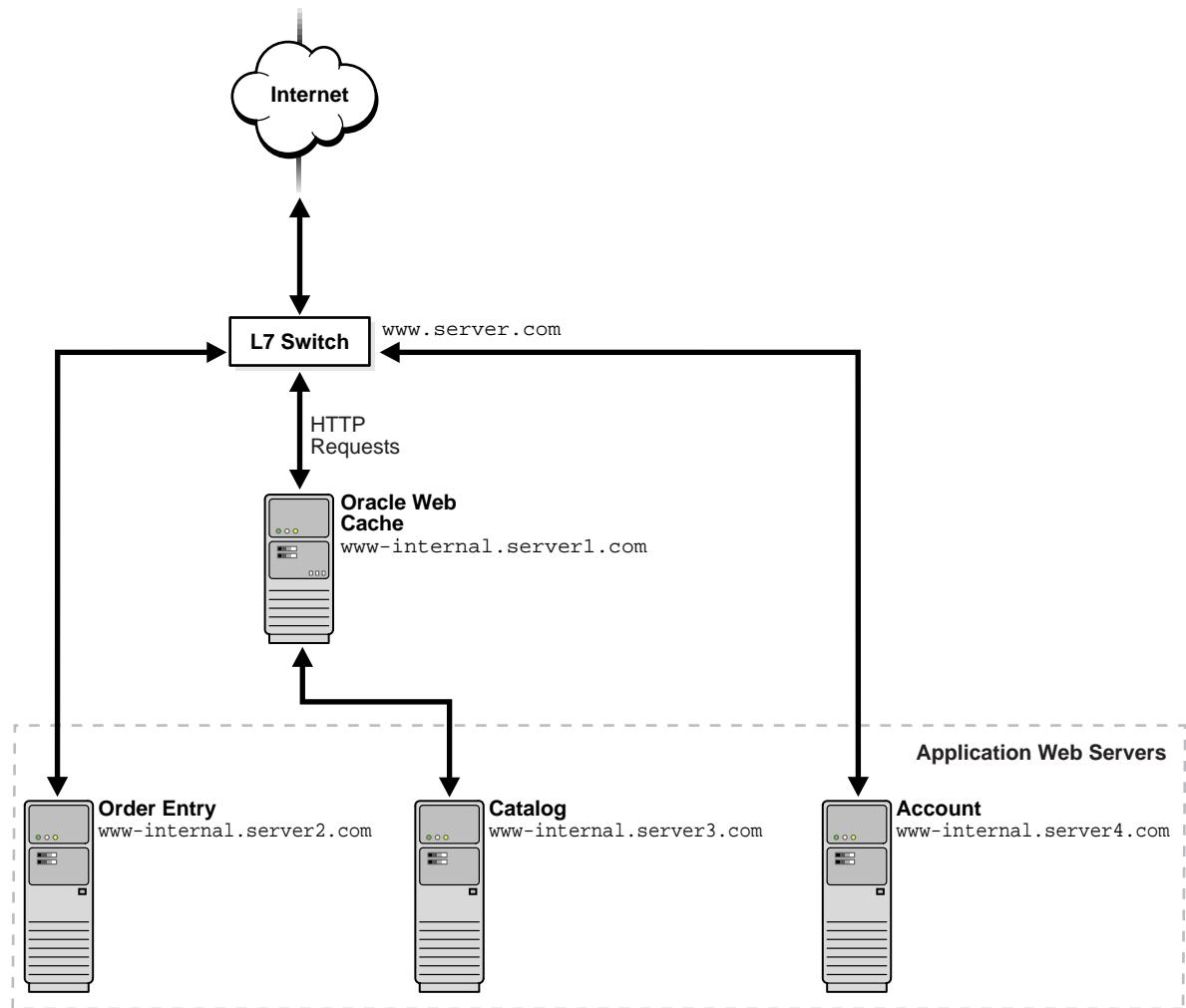
Accelerating Portions of a Web Site

Many Web sites contain cacheable catalog content and non noncacheable secure and search content. For these Web sites, you can use Oracle Web Cache servers to cache content for just the portions of the Web site with the cacheable content. [Figure 3-6](#) on page 3-9 shows an **Layer 7 (L7) switch** passing catalog requests to Oracle Web Cache server `www-internal.server1.com` and order entry and account requests to application Web servers `www-internal.server2.com` and `www-internal.server4.com`.

An L7 switch operates at Layer 7, the Application Layer layer, of the OSI model. L7 switches determine where to send requests based on URL content.

See Also: <http://www.ietf.org/> for information about the OSI stack

Figure 3–6 Accelerating Portions of a Web Site



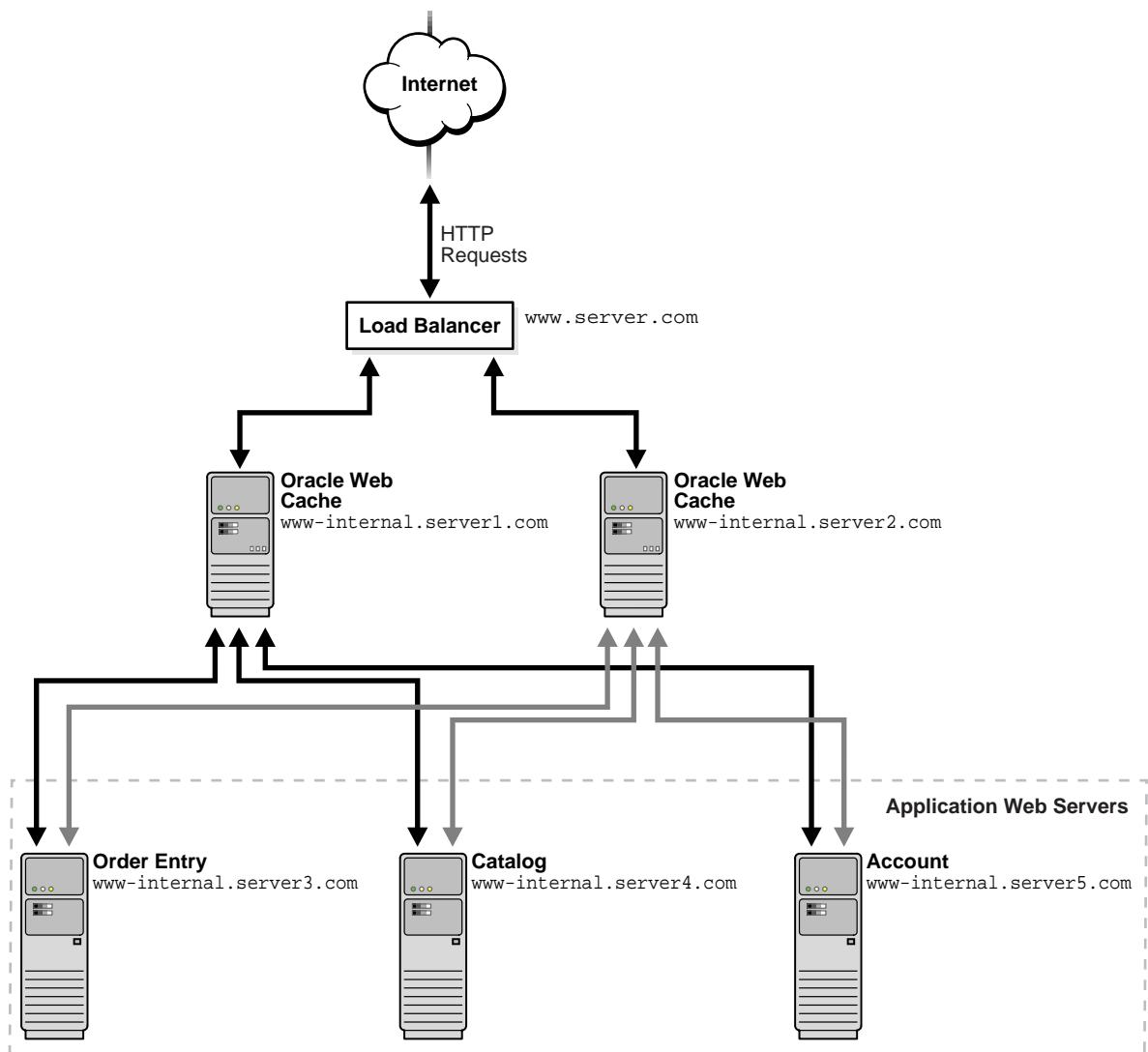
To configure this deployment:

1. Configure the L7 switch with the host name of the Oracle Web Cache server.
2. Configure Oracle Web Cache with the host names of the application Web servers for which it is caching documents. In this example, Oracle Web Cache server `www-internal.server1.com` is configured to cache for application Web server `www-internal.server3.com`.

Using Oracle Web Cache Servers in a Failover Pair

To maintain performance during an application Web server failure, you can configure two Oracle Web Cache servers as a failover pair. Both Oracle Web Cache servers are configured to cache the same content. When both Oracle Web Cache servers are running, a Load Balancer distributes the load among both servers. If one server fails, the other server receives and processes all incoming requests. This deployment is depicted in [Figure 3-7](#).

Figure 3-7 Configuring Multiple Oracle Web Caches as a Failover Pair



To configure this deployment:

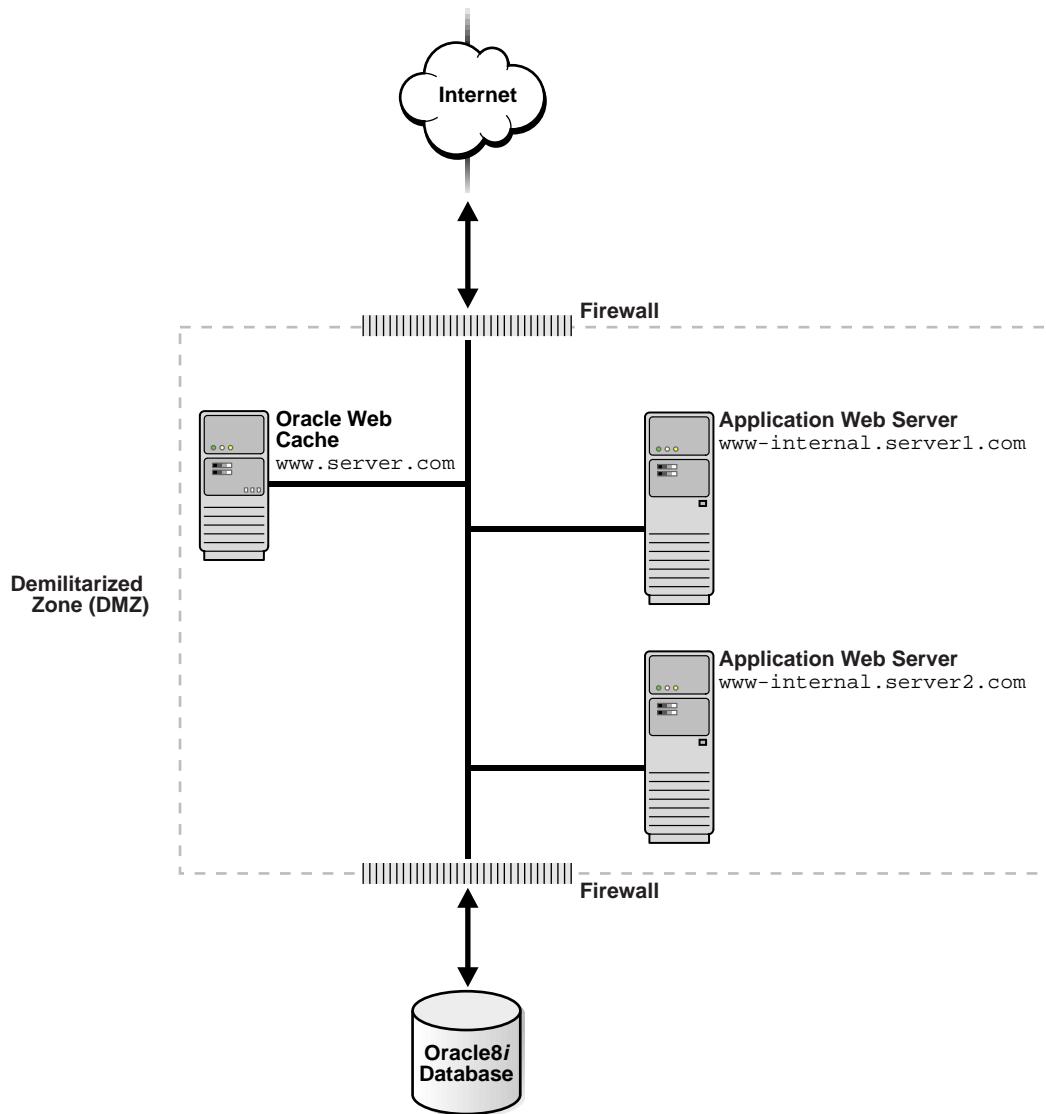
1. Configure the Load Balancer with the host names of the Oracle Web Cache servers.
2. Configure each Oracle Web Cache server with the host names of the application Web servers.

Working with Firewalls

You can deploy Oracle Web Cache inside or outside a firewall.

[Figure 3–8](#) on page 3-12 shows Oracle Web Cache positioned inside a firewall. Deploying Oracle Web Cache inside a firewall ensures that HTTP traffic enters the Demilitarized Zone (DMZ), but only authorized traffic from the application Web servers can directly interact with the database.

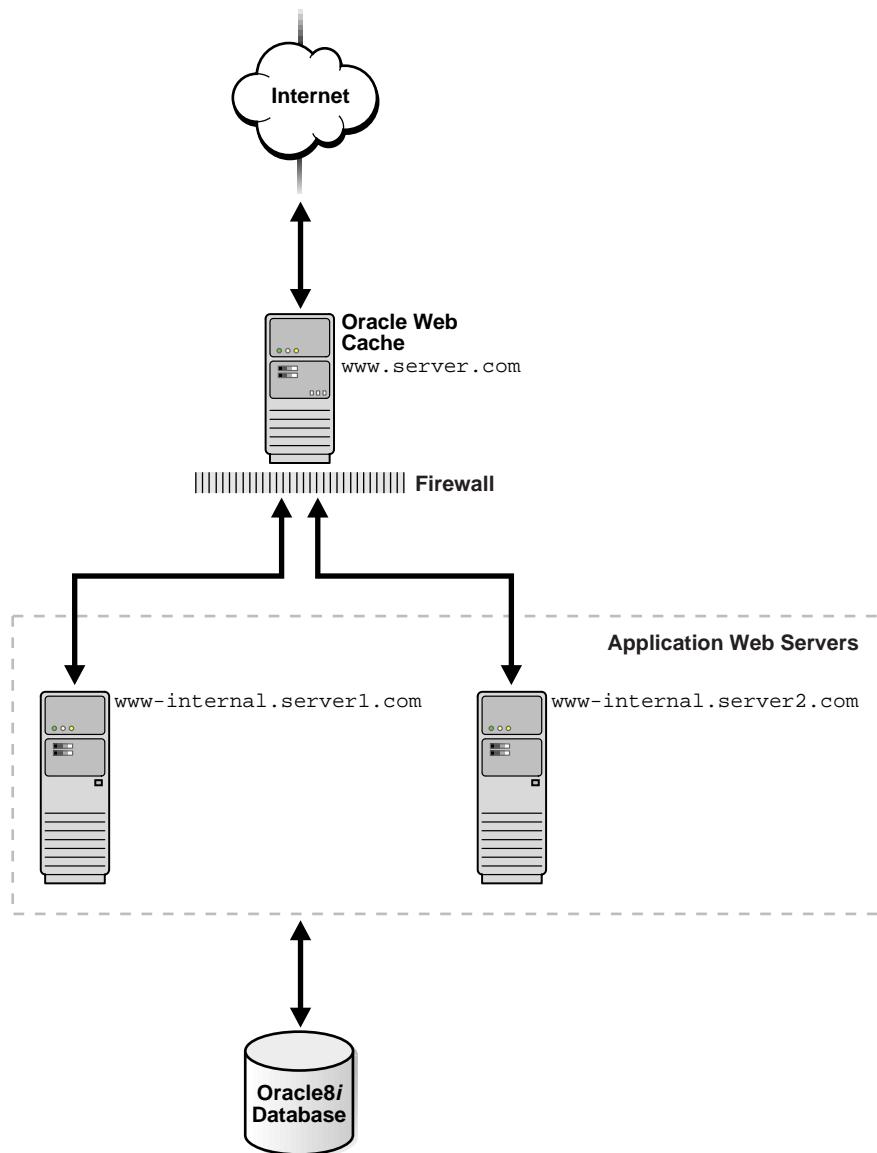
Figure 3–8 Configuring Oracle Web Cache Inside a Firewall



[Figure 3–9](#) on page 3-13 shows Oracle Web Cache positioned outside a firewall. With this deployment, the throughput burden is placed on Oracle Web Cache rather than the firewall. The firewall receives only requests that must go to the application Web servers. This deployment requires securing Oracle Web Cache from intruders.

Security experts disagree about whether caches should be placed outside the DMZ. Oracle Corporation recommends that you check your company's policy before deploying Oracle Web Cache outside the DMZ.

Figure 3-9 Configuring Oracle Web Cache Outside a Firewall

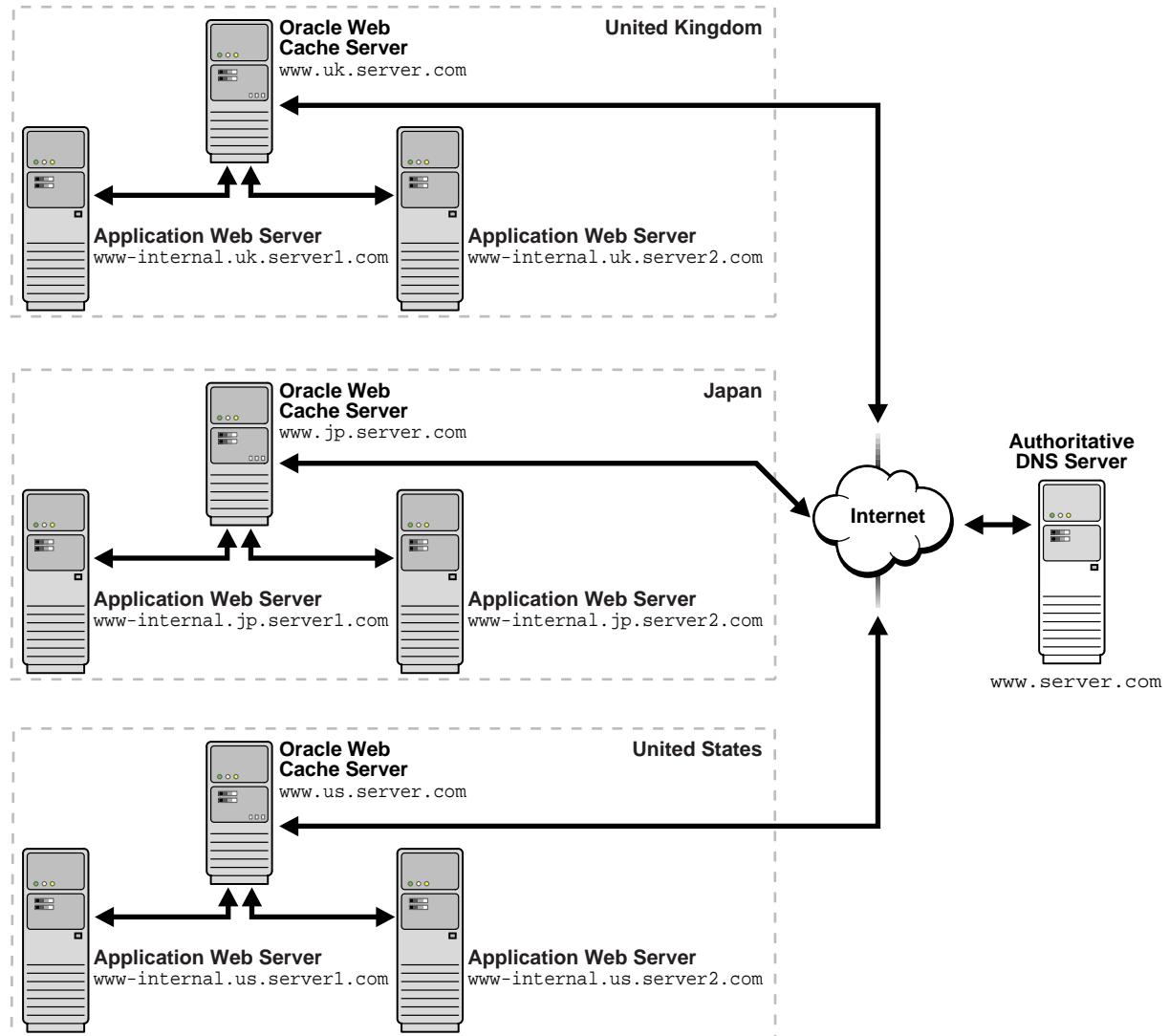


Deploying Oracle Web Cache Servers in a Distributed Network

Many Web sites have several data centers. For networks with a distributed topology, you can deploy Oracle Web Cache at each of the data centers.

[Figure 3-10](#) on page 3-15 shows a distributed topology in which Oracle Web Cache is distributed in offices in the United Kingdom, Japan, and the United States.

Browsers make a request to local DNS servers to resolve `www.server.com`. The local DNS server is routed to the authoritative DNS server `www.server.com`. The authoritative DNS server uses the IP address of the browser, the topology model, and the current load to pick an Oracle Web Cache server to satisfy the request. It then returns the IP address of the appropriate Oracle Web Cache server to the browser.

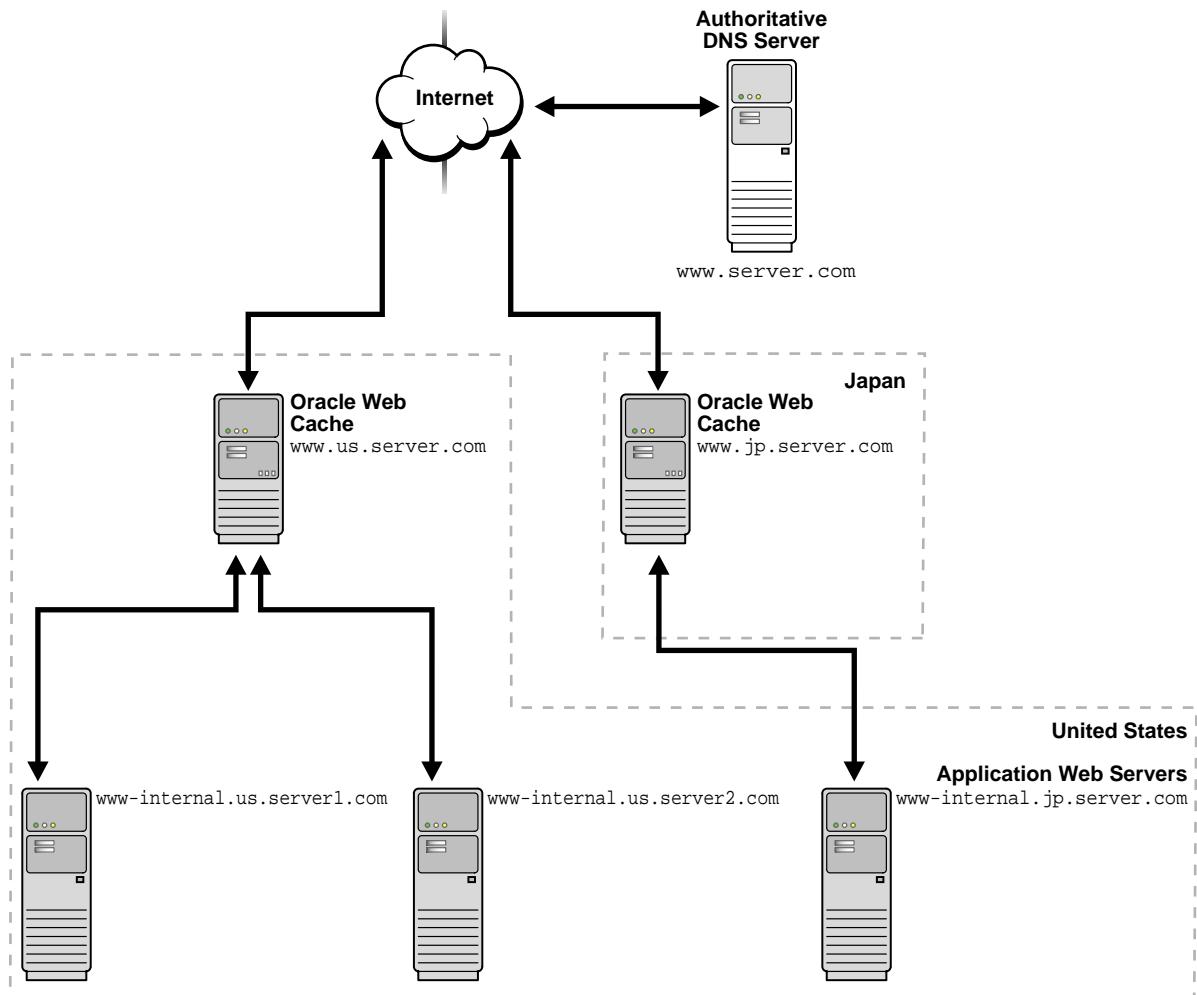
Figure 3–10 Distributed Caching

To configure this deployment:

1. Configure the local DNS servers with the location of the authoritative DNS server.
2. Configure the authoritative DNS server with the host names and IP addresses of the Oracle Web Cache servers throughout the distributed network.
3. Configure each Oracle Web Cache server with the host names of the application Web servers it is caching documents for.

Another distributed deployment solution is depicted in [Figure 3-11](#). In this deployment, an Oracle Web Cache server is located in the United States office and another server is located in the Japan office. The application Web servers for both offices are located in the United States office, centralizing the data source to one geographic location.

Figure 3–11 Centralizing the Data Source



To configure this deployment, configure each Oracle Web Cache server with the host names of the application Web servers for which it is caching documents. In this example:

- Oracle Web Cache server `www.us.server.com` is configured to cache for application Web servers `www-internal.us.server1.com` and `www-internal.us.server2.com`
- Oracle Web Cache server `www.jp.server.com` is configured to cache for application Web servers `www-internal.jp.server.com`.

4

Configuration and Administration Tools Overview

This chapter introduces the various administration tools of Oracle Web Cache. It discusses the main administration application and tells you how to launch it and navigate through it. It also introduces the command line tool.

This chapter contains these topics:

- [Oracle Web Cache Manager](#)
- [webcachectl Utility](#)
- [Configuration and Administration Tasks at a Glance](#)

Oracle Web Cache Manager

Oracle Web Cache Manager is a graphical user interface tool that combines configuration abilities with administration to provide an integrated environment for configuring and managing Oracle Web Cache.

This section introduces you to the features of Oracle Web Cache Manager. However, the primary documentation for using Oracle Web Cache Manager is the accompanying online help. This section contains these topics:

- [Starting Oracle Web Cache Manager](#)
- [Navigating Oracle Web Cache](#)

Starting Oracle Web Cache Manager

To start Oracle Web Cache Manager:

1. Ensure that the `admin` server process is started.

See Also: ["Starting and Stopping Oracle Web Cache"](#) on page 8-2

2. Point your browser to the following URL:

`http://web_cache_hostname:4000/webcacheadmin`

3. When prompted for the administrator user ID and password, enter `administrator` for the user name, and enter the appropriate password. The first time you log in, the password is `administrator`.

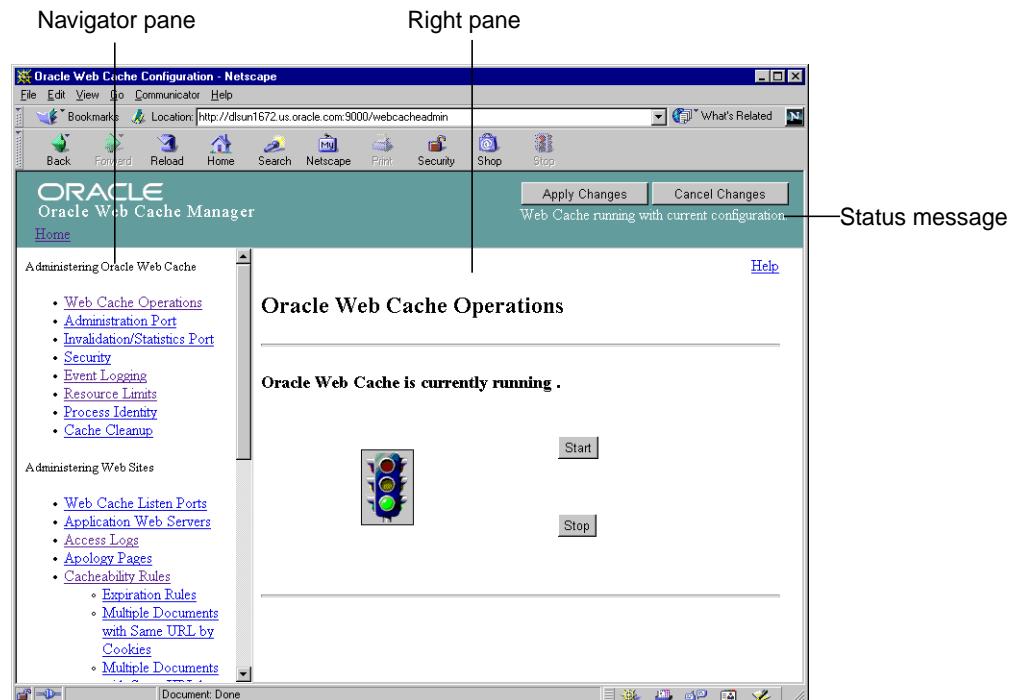
Note: You can also point your browser to `http://web_cache_hostname:4000` to link to Oracle Web Cache Manager, various README files, user documentation, and the Oracle Technology Network.

Navigating Oracle Web Cache

The Oracle Web Cache Manager interface includes:

- Top menu bar containing **Apply Changes** and **Cancel Changes** buttons and Oracle Web Cache status message
- Navigator pane with configuration and monitoring menu items
- Right pane with property sheet for selected menu item

Figure 4–1 Oracle Web Cache Configuration Interface



Apply Changes and Cancel Changes Buttons

The **Apply Changes** button applies submitted configuration changes to Oracle Web Cache. The **Cancel Changes** button cancels submitted configuration changes to Oracle Web Cache.

Note: Applied configuration changes require stopping and then restarting Oracle Web Cache. See "[Starting and Stopping Oracle Web Cache](#)" on page 8-2 for further information.

Status Messages

Status messages appear below the **Apply Changes** and **Cancel Changes** buttons. [Table 4-1](#) describes the possible status messages.

Table 4-1 Oracle Web Cache Manager Status Messages

Message	Description
Web Cache running with current configuration.	This message appears if Oracle Web Cache is running with an up-to-date configuration.
Press "Apply Changes" to commit your modifications.	This message appears if Submit has been selected in some dialog box, but the Apply Changes button has not been chosen.
Restart Oracle Web Cache to make configuration changes take effect	This message appears if Oracle Web Cache is running with an older version of the configuration. This can happen if configuration changes have been applied but Oracle Web Cache has not been restarted.

Navigator Pane

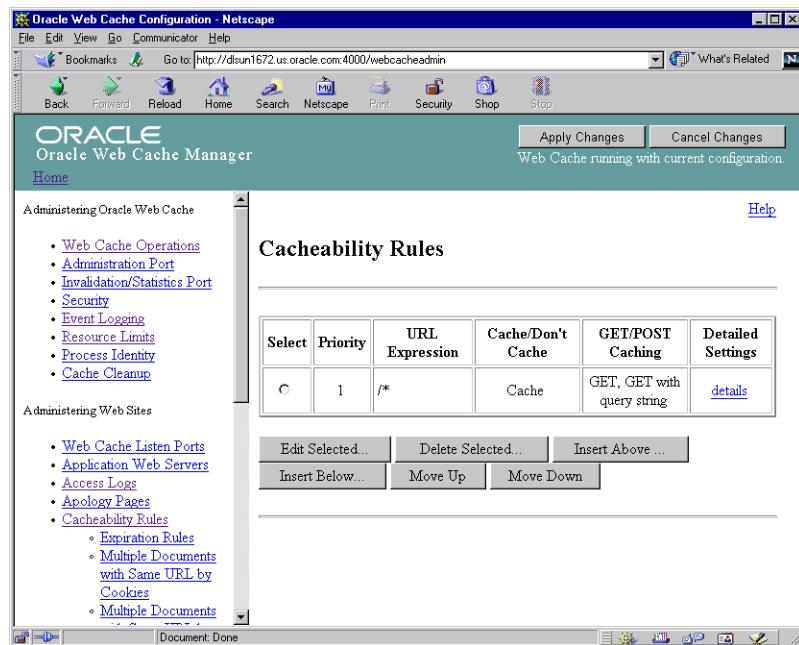
The navigator pane provides a graphical tree view of configuration, administration, and performance monitoring capabilities for Oracle Web Cache and its supported Web sites. The navigator pane contains the following major folders:

Administering Oracle Web Cache	Contains pages that enable you to: <ul style="list-style-type: none">■ Start and stop Oracle Web Cache■ Configure listening ports for administration and invalidation requests■ Configure event logging settings■ Specify the storage size of the cache■ Invalidate documents in the cache
Administering Web Sites	Contains pages that enable you to: <ul style="list-style-type: none">■ Specify the Web sites and the application Web servers that Oracle Web Cache will cache documents for■ Configure access logging settings■ Configure session tracking settings■ Configure cacheability rules■ Configure compression
Monitoring Oracle Web Cache	Contains pages that enable you to monitor the performance of Oracle Web Cache
Monitoring Application Web Servers	Contains pages that enable you to monitor the performance of application Web servers

Right Pane

The right pane contains property sheets that enable you to configure and administer Oracle Web Cache. [Figure 4–2](#) shows the Cacheability Rules property sheet used for viewing cacheability rules.

Figure 4–2 Cacheability Rules Property Sheet



webcachectl Utility

The `webcachectl` utility enables you to administer Oracle Web Cache. The general syntax for this utility follows:

```
webcachectl command
```

The possible commands for the `webcachectl` utility are `start` to start Oracle Web Cache, `stop` to stop Oracle Web Cache, and `status` to obtain the current status of Oracle Web Cache. For example, the following command starts Oracle Web Cache:

```
webcachectl start
```

See Also: ["Starting and Stopping Oracle Web Cache" on page 8-2](#)

Configuration and Administration Tasks at a Glance

Oracle Web Cache configuration and administration tasks are described throughout this guide and in the Oracle Web Cache Manager online help system. **Table 4–2** lists the common tasks, and points you to the topic in this guide that describes the task.

Table 4–2 Common Administrative Tasks for Oracle Web Cache

Task	See Also
Configuring Oracle Web Cache	
Change the administrator's password.	" Task 2: Modify Security Settings " on page 5-2
Set the maximum cache size limit.	" Task 3: Set Resource Limits " on page 5-5
Configure support for a Web site.	" Task 4: Specify Web Site Settings " on page 5-7
Configure cacheability rules.	" Configuring Cacheability Rules " on page 6-5
Load balance requests over multiple application Web servers.	" Configuring Load Balancing " on page 7-2
Bind a session to an application Web server.	" Binding a Session to an Application Web Server " on page 7-3
Configure event log settings.	" Configuring Event Logs " on page 8-21
Configure access log settings.	" Configuring Access Logs " on page 8-27
Administering Oracle Web Cache	
Start and stop Oracle Web Cache	" Starting and Stopping Oracle Web Cache " on page 8-2
Invalidate documents in the cache.	" Invalidate Documents in the Cache " on page 8-4
Monitoring Performance	
Monitor Oracle Web Cache overall health.	" Monitoring Overall Cache Health " on page 9-3
Monitor Oracle Web Cache performance.	" Gathering Oracle Web Cache Performance Statistics " on page 9-5
Monitor application Web server performance.	" Gathering Application Web Server Performance Statistics " on page 9-6

Note: All tasks listed under the heading **Configuring Oracle Web Cache** row require stopping and then restarting Oracle Web Cache. See "[Starting and Stopping Oracle Web Cache](#)" on page 8-2 for further information.

5

Initial Setup and Configuration

This chapter describes the steps to initially configure Oracle Web Cache to begin caching application Web server content after installation.

This chapter contains these topics:

- [Task 1: Start Oracle Web Cache](#)
- [Task 2: Modify Security Settings](#)
- [Task 3: Set Resource Limits](#)
- [Task 4: Specify Web Site Settings](#)
- [Task 5: Specify Caching Rules](#)
- [Task 6: Restart Oracle Web Cache](#)

Task 1: Start Oracle Web Cache

To start Oracle Web Cache to begin initial configuration:

1. If not currently logged on to the Oracle Web Cache computer, log in with the user ID of the user that performed the installation.
2. Start Oracle Web Cache. From the command line, enter:

```
webcachectl start
```

Task 2: Modify Security Settings

When Oracle Web Cache is first installed, it is set up with default passwords for administration and invalidation. In addition, the computer on which you installed Oracle Web Cache is the default trusted host.

To change the security settings:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager" on page 4-2](#)

2. Change the password for the administrator. Configuration and operational tasks can be performed with the Oracle Web Cache administrator user. The administrator user has a default password of `administrator` set up during installation. Before you begin configuration, change the default password to a secure password.
 - a. In the navigator pane, select **Administering Oracle Web Cache > Security**.
The Security page appears in the right pane.
 - b. In the Security page, choose **Change Admin Password** under **Administration User**.
The Change Administration User Password dialog box appears.
 - c. Enter `administrator` in the **Old Password** field and a new password between four and 10 characters in the **New Password** and **Confirm New Password** fields.
 - d. Choose **Submit**.

3. Optionally, change the password for the invalidation administrator. The invalidation administrator has a user ID of `invalidator`, whose default password of `invalidator` is set up during installation.
 - a. In the Security page, choose **Change Invalidation Password** under the **Invalidation User**.

The Change Invalidation User Password dialog box appears.
 - b. Enter `invalidator` in the **Old Password** field, and a new password between four and 10 characters in the **New Password** and **Confirm New Password** fields.
 - c. Choose **Submit**.
4. Optionally, change the trusted subnet or trusted host from which Oracle Web Cache and invalidation administration can take place. By default, the computer on which you installed Oracle Web Cache is the trusted host.
 - a. In the Security page, choose **Change Trusted Subnets** under the **Currently trusted subnets**.

The Change Trusted Subnets dialog box appears.
 - b. Select one of the following options:
 - All subnets**

Select to allow administration requests from all computers in all the subnets in the network.
 - This machine only**

Select to allow administration and invalidation requests from only this computer.
 - Enter list of IPs**

Select to allow administration and invalidation requests from all IP addresses you enter in a comma-separated list.
 - c. Choose **Submit**.

5. Optionally, change the user ID and group ID for the Oracle Web Cache executables on UNIX. By default, the user that performed the installation is the owner of Oracle Web Cache executables. Only this user can execute `webcachectl start|stop` commands.
 - a. In the navigator pane, select **Administering Oracle Web Cache > Process Identity**.
The Process Identity page appears in the right pane.
 - b. In the Process Identity page, choose **Change IDs**.
The Change Process Identity dialog box appears.
 - c. Enter the new user in the **New User ID** field and the group ID of the user in the **New Group ID** field.
 - d. Choose **Submit**.
6. In the Oracle Web Cache Manager main window, choose **Apply Changes**.

Task 3: Set Resource Limits

When the maximum cache memory limit is reached, Oracle Web Cache performs garbage collection. During garbage collection, Oracle Web Cache removes the less popular and less valid documents from the cache in favor of the more popular and more valid documents.

To avoid swapping documents in and out of the cache, it is crucial to configure enough memory for the cache. By default, the memory limit is set to 500 MB, which is sufficient for most caches. To acquire a rough estimate of the memory required for the cache, use the following formula:

(average HTTP object size) * (maximum number of objects you want to cache)

For example, if you want to cache 10,000 objects and the average the size of those objects is 3 KB, then the maximum cache size limit should be set to at least 30 MB.

When setting the maximum memory usage limit, Oracle Corporation recommends setting the limit as close to the operating system's resource limit as possible. If necessary, resize the operating system's resource limit.

Note: Most operating systems have a 2 GB memory limit.

The size and maximum number of Web objects, such as GIF, HTML, or PDF, can be determined by:

- Application Web server's access log files
- Simple shell scripts
- Third-party tools

In addition to the cache size, it is also important to specify the expected load on the Oracle Web Cache server. The expected load is the sum of the maximum number of incoming open connections to the Oracle Web Cache server and the number of outgoing open connections to the application Web servers. When you configure this limit, set a reasonable number. If you set a number that is too high, performance can be affected. To help you determine the number, you can use various tools available for your operating system. For example, the `netstat` command on UNIX enables you to determine the number of established connections.

Note: Sun Solaris release 2.6 can have no more than 1,000 connections. Sun Solaris release 2.7 or higher can have up to 65,000.

See Also:

- Operating system specific documentation for connection limitations
- *Oracle Internet Application Server 8i Oracle HTTP Server powered by Apache Performance Guide* for TCP/IP performance tuning tips

To set resource limits:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager" on page 4-2](#)

2. Set the maximum cache size:
 - a. In the navigator pane, select **Administering Oracle Web Cache > Resource Limits**.
The Resource Limits page appears in the right pane.
 - b. In the Resource Limits page, choose **Change cache size limit**.
The Change Maximum Cache Size dialog box appears.
 - c. In the **New maximum cache size**, enter the new cache size.
 - d. Choose **Submit**.
3. Set the maximum incoming connections:
 - a. In the Resource Limits page, choose **Change connections limit**.
The Change Maximum Incoming Connections Limit dialog box appears.
 - b. In the **New maximum connections limit** field, enter the new limit.
 - c. Choose **Submit**.
4. In the Oracle Web Cache Manager main window, choose **Apply Changes**.

Task 4: Specify Web Site Settings

For Oracle Web Cache to act as a virtual server for a Web site, configure Oracle Web Cache with information about the Web site, including the host names of the application Web servers. In addition, specify a listening port from which Oracle Web Cache can receive browser requests.

To configure Oracle Web Cache with Web site information:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager"](#) on page 4-2

2. Configure the application Web servers for the Web site:

- a. In the navigator pane, select **Administering Web Sites > Application Web Servers**.

The Application Web Servers page appears in the right pane.

- b. In the Application Web Servers page, choose **Add**.

The Edit/Create Application Web Server page dialog box appears.

- c. In the **Hostname** field, enter the host name of the application Web server.

- d. In the **Port** field, enter the listening port from which the application Web server will receive Oracle Web Cache requests.

- e. In the **Capacity** field, enter the number of concurrent connections that the application Web server can sustain.

When you set the capacity, it assigns a weighted load percentage to the application Web server. The load specifies the percentage of requests that this application Web server will handle. The load percentage is calculated from the following formula:

$$\text{application Web server capacity} / \text{combined capacity}$$

For example, if one application Web server has a capacity of 50 and a second application Web server has a capacity of 40 for a total capacity of 90, then the first server is assigned a load percentage of 55 and the second server is assigned a load percentage of 45.

$$\begin{aligned} 50/90 &= 55\% \\ 40/90 &= 45\% \end{aligned}$$

If this is the only application Web server, the load will be 100 regardless of the capacity.

The maximum number of concurrent connections that an application Web server can handle is determined by load testing the application Web server until it runs out of CPU, responds slowly, or until a backend database reaches full capacity.

- f. In the **Failover Threshold** field, enter the number of allowed continuous request failures before Oracle Web Cache will consider the application Web server down. The default is five requests.

If an application Web server fails any time after Oracle Web Cache has started to send a request, then Oracle Web Cache increments the failure counter. The failure counter is reset in the event of a successful application Web server response. A request is considered failed if:

- There are any network errors
- The HTTP response status code is either less than 100, or is one of the 500 (Internal Server Error), 502 (Bad Gateway), 503 (Service Unavailable), or 504 (Gateway Timeout) messages

Once the threshold is met, Oracle Web Cache considers the application Web server down and uses other application Web servers for future requests. When an application Web server is down, Oracle Web Cache starts polling the application Web server. It does this by sending requests to the URL specified in the **Ping URL** field. When Oracle Web Cache is able to successfully get a response from the application Web server without any network errors and the HTTP response code is not less than 100, or equal to 500, 502, 503, 504, it considers that application Web server live again and uses it for future requests.

Note: The threshold does not apply if Oracle Web Cache cannot connect to an application Web server. In this case, Oracle Web Cache immediately considers the application Web Cache down and does not use it for future requests. The failover to other live application Web servers does not apply if there is only one live application Web server left.

- g. In the **Ping URL** field, enter the URL that Oracle Web Cache will use to poll an application Web server that has reached its failover threshold.
 - h. In the **Ping Interval (seconds)** field, enter the time in seconds that Oracle Web Cache will poll an application Web server that has reached its failover threshold. The default is 10 seconds.
 - i. Choose **Submit**.
3. Optionally, configure an additional listening port from which Oracle Web Cache will receive browser requests.

Oracle Web Cache listens on port 1100 by default. It may be necessary to add an additional listening port if you want to assign Oracle Web Cache a port that an application Web server was previously listening on.

 - a. In the navigator pane, select **Administering Web Sites > Oracle Web Cache Listen Ports**.

The Oracle Web Cache Listen Ports page appears in the right pane.
 - b. In the Oracle Web Cache Listen Ports page, choose **Add**.

The Edit/Create Oracle Web Cache Listen Ports page dialog box appears.
 - c. In the **Oracle Web Cache IP Address** field, enter the IP address of the computer running Oracle Web Cache.
 - d. In the **Oracle Web Cache Listen Port** field, enter the listening port from which Oracle Web Cache will receive Web browser requests for the Web site. Ensure this port number is not already in use.
 - e. Choose **Submit**.
 4. In the Oracle Web Cache Manager main window, choose **Apply Changes**.

Task 5: Specify Caching Rules

Specify the URLs containing the documents you want Oracle Web Cache to cache.

See Also: "Configuring Cacheability Rules" on page 6-5

Task 6: Restart Oracle Web Cache

When Oracle Web Cache is configured, stop it and start it again to read in the new configuration settings. You can stop and start Oracle Web Cache using either Oracle Web Cache Manager or the `webcachectl` utility on the computer on which Oracle Web Cache software is installed and configured:

Use Oracle Web Cache Manager...	Use the <code>webcachectl</code> Utility...
<ol style="list-style-type: none">1. Start Oracle Web Cache Manager. See Also: "Starting Oracle Web Cache Manager" on page 4-22. In the navigator pane, select Administering Oracle Web Cache > Web Cache Operations. The Oracle Web Cache Operations page appears in the right pane.3. In the Oracle Web Cache Operations page, choose Stop and then Start.	<p>From the command line, enter:</p> <pre>webcachectl stop webcachectl start</pre>

6

Creating Rules for Cached Content

This chapter explains how to configure cacheability rules. It contains these topics:

- [Cacheability Rules Overview](#)
- [Configuring Cacheability Rules](#)
- [Configuring Expiration Rules](#)
- [Configuring Rules for Multiple-Version URLs Containing Cookies](#)
- [Configuring Rules for Multiple-Version URLs Containing HTTP Request Headers](#)
- [Configuring Rules for Personalized Pages](#)
- [Configuring Rules for Pages with Session Tracking](#)

Cacheability Rules Overview

Using Oracle Web Cache to specify cacheability rules, you can select to cache or not to cache content for static documents, multiple-version **URLs**, personalized pages, pages that support session tracking, and HTTP error messages.

Generally, when you assign cacheability rules, you specify the **regular expression** matching the URL and whether you want the documents contained in the URL cached or not cached. You then order the cacheability rules in order of priority. Higher priority rules are processed first.

For cacheable regular expressions that contain a document or a subset of documents that are not cacheable, give the non-cacheable documents a higher priority than the cacheable documents.

For example, if you want all URLs containing

/cec/cstage?ecaction=ecpassthru to be cached except for
/cec/cstage?ecaction=ecpassthru2, you would enter the rules in the following order:

1. ^/cec/cstage\?ecaction=ecpassthru2 (Don't Cache)
2. ^/cec/cstage\?ecaction=ecpassthru.* (Cache)

If the order were reversed, all documents starting with

/cec/cstage?ecaction=ecpassthru would be cached, including
/cec/cstage?ecaction=ecpassthru2.

Examples of content that administrators would typically declare non-cacheable include update transactions, shopping cart views, personal account views, and so forth. One of the easiest ways to set up cacheability rules in Oracle Web Cache is either to first specify the non-cacheable content, and then use a broad "catch-all" rule for the cacheable content, or to first specify the cacheable content followed by a non-cacheable catch-all rule. In practice, cacheable and non-cacheable rules can be interspersed.

If no cacheability rules are specified, then Oracle Web Cache behaves just as HTTP proxy cache does, that is, it relies on HTTP header information to determine what is cacheable. Generally, HTTP proxy caches store only pages with static content.

Cacheability Rule Syntax

Please note that cacheability rules use regular expression syntax, which is based on the POSIX 1003 extended regular expressions for URLs, as supported by Netscape Proxy Server 2.5.

When using POSIX regular expression, keep the following syntax rules in mind:

- Use a carat (^) to denote the beginning and a dollar sign (\$) to denote the end of the URL.

If these characters are not used, POSIX assumes a substring match. For example, `^/a/b/.*\.gif$` will match GIF files under `/a/b` or any of its subdirectories. `/a/b/.*\.gif`, on the other hand, could match `/x/y/a/b/c/d.gif`.
- Use a backslash (\) to escape any special characters, such as periods (\.), question marks (\?), or asterisks (*).

See Also:

http://www.cs.utah.edu/dept/old/texinfo/regex/reg_ex_toc.html for regular expression syntax

Table 6-1 shows examples of content to cache and how to enter regular expression syntax for corresponding cacheability rules for that content.

Table 6-1 Regular Expression Examples

Content to Cache	Regular Expression Syntax
URL beginning with /machine/doc and ending in *.gif	<code>^/machine/doc/.*\.gif\$</code>
All Graphics Interchange Format (GIF) images	<code>\.gif\$</code>
/robots.txt file	<code>^/robots.txt\$</code>
All procedures in the new_employee package	<code>^/pls/enroll_db/new_employee</code>

Default Cacheability Rules

Table 6–2 shows examples of content to cache and how to enter regular expression syntax for corresponding cacheability rules for that content.

Table 6–2 Regular Expression Examples

URL Expression	Cache/Don't Cache	Description
\.pdf\$	Don't Cache	Instructs Oracle Web Cache to not cache files ending in .pdf
\.html\$	Cache	Instructs Oracle Web Cache to cache all files ending in .htm and .html

Note: Oracle Web Cache cannot cache HTTP multi-part responses to HTTP Range requests. If the application Web servers that Oracle Web Cache sends HTTP requests to return certain documents in multi-part format, configure these documents as non-cacheable. For example, certain browsers send Range requests for PDF documents; therefore, the cacheability rules for all PDF documents should be set non-cacheable.

Configuring Cacheability Rules

To configure cacheability rules:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager"](#) on page 4-2

2. In the navigator pane, select **Administering Web Sites > Cacheability Rules**.

The Cacheability Rules page appears in the right pane.

3. In the Cacheability Rules page, choose **Create** if no rules exist. If rules already exist, select a rule, and then choose **Insert Above** or **Insert Below**.

The Create Cacheability Rule or Edit/Create Cacheability Rule dialog box appears.

4. In the **URL Expression** field, enter regular expression syntax, matching the URLs to which you want the cacheability rule to apply. Remember to use "^" to denote the start of the URL and "\$" to denote the end of the URL.

5. Select **Cache** or **Don't Cache** for the documents contained within the URL.

If you select **Cache**, continue to Step 6. If you select **Don't Cache**, skip to Step 7.

6. Select options for the columns that apply:

**GET/POST
Caching**

Select or deselect to cache documents that contain GET, GET with query string, or POST **HTTP request methods** in forms.

Important: If your Web site's GET with query string or POST methods are used for forms that make changes to the application Web servers or database, do not select **Get with query string** or **POST**. These options should only be selected if the forms are used in search forms.

Expiration Rule

From the list, select an expiration rule to apply to the documents. If you do not see an expiration rule suitable for the documents, choose **Create A New Rule** to create a new rule.

See Also: Step 4 in ["Configuring Expiration Rules"](#) on page 6-12

Multiple Documents with Same URL by Cookies

Select **None** to not have Oracle Web Cache cache multiple-version documents that use cookies.

Select **Apply the following** to have Oracle Web Cache cache multiple-version documents that rely on **category cookie** values, and then select the required cookies. If you do not see a cookie rule that can be applied to these documents, choose **Create A New Rule** to create a new policy or modify an existing policy.

See Also: Step 4 in "[Configuring Rules for Multiple-Version URLs Containing Cookies](#)" on page 6-14

Multiple Documents with Same URL by Other Headers

Select the **HTTP request headers** whose values Oracle Web Cache will use to cache and identify multiple-version URLs.

Accept: Specifies which media types are acceptable for the response

Accept-Charset: Specifies which character sets are acceptable for the response

Accept-Encoding: Restricts the content-encodings that are acceptable in the response

Accept-Language: Specifies the set of languages that are preferred as a response

User-Agent: Contains information about the Web browser that initiated the request

An example of a request made with a Netscape 4.6 browser with HTTP request headers follows:

```
User-Agent: Mozilla/4.61 [en] (WinNT; U)
Accept: image/gif, image/x-bitmap, image/jpeg,
image/pjpeg, image/png,*/
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
```

Note: Oracle Web Cache does not interpret the values of these HTTP request headers. If the values for two pages are different, Oracle Web Cache caches both pages separately. For example, if one request sends a HTTP request header of User-Agent : Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0) and another request sends a HTTP request header of User-Agent : Mozilla/4.0 (compatible; MSIE 5.0; Windows NT; DigExt) for different versions of Internet Explorer, Oracle Web Cache serves two pages.

Session-Related Caching Rules	Select None to not have Oracle Web Cache cache documents that use session information contained within a session cookie or embedded in a URL as a parameter. Select Apply the following to have Oracle Web Cache cache documents with session information, and then select the required session definitions. If you do not see the session these documents require, choose Create A New Rule to create a new rule. See Also: " Configuring Rules for Pages with Session Tracking " on page 6-24 for further information about creating session-related caching rules
Personalized Pages	Select No to cache documents with personalized attributes or session-encoded URLs . Select Yes to cache documents with personalized content, and then select one of the following options: <ul style="list-style-type: none">▪ pages do not contain HREFs that are session encoded URLs to cache substitution instructions for only personalized attributes▪ pages contain HREFs that are session encoded URLs to cache substitution instructions for both personalized attributes and session-encoded URLs Important: To use the personalized attribute feature, enclose the personalized attribute information with the <!-- WEBCACHETAG> and the <!-- WEBCACHEEND--> HTML tags.
HTTP Error Caching	Enter the HTTP errors codes you want Oracle Web Cache to cache. If you enter multiple codes, use a comma to separate them. If there is a problem on the application Web servers that will remain unresolved, you can cache the error until the problem is resolved. Once the problem is resolved, you should invalidate the cached HTTP errors. See Also: " Invalidating Documents in the Cache " on page 8-4
Need Compression	Select to compress the documents upon insertion into the cache. If a document retrieved from the application Web server already contains a Content-Encoding header, which is typically used to denote compression, Oracle Web Cache will not compress it.

7. Choose **Submit**.
8. Repeat Steps 3 through 7 for each cacheability rule.
9. In the Cacheability Rules page, order the rules by their priority. Select a cacheability rule and choose **Move Up** or **Move Down** to order the rules. Higher priority rules are processed first.
10. Apply changes and restart Oracle Web Cache:
 - a. In the Oracle Web Cache Manager main window, choose **Apply Changes**.
 - b. In the navigator pane, select **Administering Oracle Web Cache > Web Cache Operations**.
The Oracle Web Cache Operations page appears in the right pane.
 - c. In the Oracle Web Cache Operations page, choose **Stop** and then **Start** to restart Oracle Web Cache.

Cacheability Rule Example

Figure 6-1 illustrates how an administrator might set up cacheability rules for a catalog built on Oracle iStore technology, which enables e-merchants to design, build, and publish their stores on the World Wide Web.

Figure 6-1 Cacheability Rules Example

The screenshot shows the Oracle Web Cache Manager interface running in Netscape Communicator. The title bar reads "Netscape: Oracle Web Cache Configuration". The main window displays a table of cacheability rules with columns: Select, Priority, URL Expression, Cache/Don't Cache, GET/POST Caching, and Detailed Settings. The table contains 12 rows of rules. The left sidebar lists navigation links for Web Cache Operations, Administration Port, Invalidation/Statistics Port, Security, Event Logging, Resource Limits, Process Identity, Cache Cleanup, Administering Web Sites, Cacheability Rules (with sub-options like Expiration Rules, Multiple Documents with Same URL by Cookies, etc.), Session Management, and Monitoring Oracle Web Cache.

Select	Priority	URL Expression	Cache/Don't Cache	GET/POST Caching	Detailed Settings
()	1	template=walkin1\en\htm	Cache	GET, GET with query string	details
()	2	ecpassthru2	Don't Cache	N/A	N/A
()	3	ecpassthru	Cache	GET, GET with query string	details
()	4	\.jpg\$	Cache	GET, GET with query string	details
()	5	\.gif\$	Cache	GET, GET with query string	details
()	6	\.html?\$	Cache	GET, GET with query string	details
()	7	ecprdtmsearch	Cache	GET, GET with query string	details
()	8	eclogout	Don't Cache	N/A	N/A
()	9	eccartview	Don't Cache	N/A	N/A
()	10	ecactroot	Don't Cache	N/A	N/A
()	11	\.pdf\$	Don't Cache	N/A	N/A
()	12	*	Cache	GET, GET with query string	details

The rules are described in [Table 6–3](#).

Table 6–3 Regular Expression Examples

Priority Order	URL Expression	Cache/Don't Cache	GET/POST Caching	Description
1	template=walkin\.\en\.\htm	Cache	GET, GET with query string	Caches the home page, including personalized information
2	ecpassthru2	Don't Cache	Not Applicable	Does not cache this template, because it contains customized user account information
3	ecpassthru	Cache	GET, GET with query string	Caches this template, because it does not contain customized content.
4	\.jpe?g\$	Cache	GET, GET with query string	Caches all JPEG images
5	\.gif\$	Cache	GET, GET with query string	Caches all GIF images
6	\.html? \$	Cache	GET, GET with query string	Caches all pages ending in .htm and .html
7	ecprditmsearch	Cache	GET, GET with query string	Caches the search results
8	eclogout	Don't Cache	Not Applicable	Does not cache logout results
9	eccartview	Don't Cache	Not Applicable	Does not cache the shopping cart
10	ecacctroot	Don't Cache	Not Applicable	Does not cache the account view
11	\.pdf\$	Don't Cache	Not Applicable	Does not cache pages ending in .pdf
12	.*	Cache	GET, GET with query string	Caches everything else in the Web site

Note: Implementations of Oracle iStore can be customized. Therefore, these cacheability rules will not apply in all Oracle iStore deployments.

Configuring Expiration Rules

You can create rules for when to expire documents in the cache. In addition, you can specify how long documents can reside in the cache once they have expired. When a document expires, it is either immediately invalidated or invalidated based on when the application Web servers can refresh them.

To create expiration rules:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager" on page 4-2](#)

2. In the navigator pane, select **Administering Web Sites > Cacheability Rules > Expiration Rules**.

The Expiration Rules page appears in the right pane.

3. In the Expiration Rules page, choose **Add**.

The Create Expiration Rule dialog box appears.

4. In the **Expire** section, specify when to expire documents by selecting one of the following options. While the first two options enable you to set expiration for Oracle Web Cache-specific rules, the third option recognizes the expiration policy established for the documents already programmed with an HTTP `Expires` header.

Expire <time> after cache entry	Select this option to base expiration on when the documents entered the cache. Enter the number of seconds to expire the documents.
--	---

Expire <time> after document creation	Select this option to base expiration on when the documents were created. Enter the number of seconds to expire the documents.
--	--

Expires as per HTTP Expires header	Select this option to respect the HTTP <code>Expires</code> header. This is the default. In order to utilize this option, documents must be programmed to use the HTTP <code>Expires</code> header.
---	---

-
5. In the **After Expiration** section, specify how you want Oracle Web Cache to process documents once they have expired.

**Remove
immediately**

Select this option to have Oracle Web Cache mark documents as invalid and then remove them immediately. A document is refreshed from the application Web server when the cache receives the next request for it.

Note: This is equivalent to a validity level of 0.

**Refresh on
demand as
application Web
server capacity
permits AND no
later than <time>
after expiration**

Select this option to have Oracle Web Cache mark documents as invalid and then refresh them based on application Web server capacity. Enter the maximum time in which the documents can reside in the cache.

Optionally, select a validity level for the documents after they expire. Validity determines how long Oracle Web Cache will serve documents stale from the cache before marking them as invalid.

The validity level ranges from 1 (for the least valid) to 9 (for the most valid). The higher the validity level, the longer Oracle Web Cache serves documents stale from the cache before marking them as invalid. Oracle Web Cache serves documents with a low validity level for a short amount of time before marking them as invalid.

Note: Performance assurance heuristics apply when you configure documents to be refreshed based on when the application Web servers can refresh them; performance assurance heuristics do not apply when documents are immediately removed.

6. Choose **Submit**.
7. Repeat Steps 3 through 6 for each expiration rule.
8. In the Expiration Rules page, choose the newly-create rule, and choose **Change URL Association**.

- The Change Policy-URL Association dialog box appears.
9. Select a URL from the right list, and then choose the **Make Association** button. The selected URL moves to the left list.
 10. Apply changes and restart Oracle Web Cache:
 - a. In the Oracle Web Cache Manager main window, choose **Apply Changes**.
 - b. In the navigator pane, select **Administering Oracle Web Cache > Web Cache Operations**.
The Oracle Web Cache Operations page appears in the right pane.
 - c. In the Oracle Web Cache Operations page, choose **Stop** and then **Start** to restart Oracle Web Cache.

Configuring Rules for Multiple-Version URLs Containing Cookies

See Also: ["Multiple Versions of the Same URL"](#) on page 2-6 for an overview and an example scenario

You can specify which category cookies whose values Oracle Web Cache will use to cache and identify multiple-version URLs.

To specify cookie values for multiple-version URLs:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager"](#) on page 4-2

2. In the navigator pane, select **Administering Web Sites > Caching Rules > Multiple Documents with Same URL by Cookies**.

The Multiversion URLs with the Same URL by Cookies page appears in the right pane.

3. In the Multiversion URLs with the Same URL by Cookies page, choose **Create**.
The Edit/Create Multiple Documents with Same URL by Cookies Rule dialog box appears.
4. In the **Cookie Name** field, enter the name of the cookie.
5. Choose **Yes** in the **Also cache documents whose requests do not contain this cookie?** prompt to cache versions of the URL that do not use this cookie. This

option enables you to serve documents from the cache for browser requests that do not use this cookie's value.

Choose **No** to not cache versions of documents that do not use this cookie's value.

6. Choose Change URL Association.

The Change Policy-URL Association dialog box appears.

7. Select a URL from the right list, and then choose the Make Association button.
The selected URL moves to the left list.

8. In the Edit/Create Multiple Documents with Same URL by Cookies Rule dialog box, choose Submit.

9. Repeat Steps 3 through 8 for each rule.

10. Apply changes and restart Oracle Web Cache:

a. In the Oracle Web Cache Manager main window, choose Apply Changes.

b. In the navigator pane, select Administering Oracle Web Cache > Web Cache Operations.

The Oracle Web Cache Operations page appears in the right pane.

c. In the Oracle Web Cache Operations page, choose Stop and then Start to restart Oracle Web Cache.

Configuring Rules for Multiple-Version URLs Containing HTTP Request Headers

See Also: ["Multiple Versions of the Same URL"](#) on page 2-6 for an overview and an example scenario

You can specify which HTTP request headers whose values Oracle Web Cache will use to cache and identify multiple-version URLs. If a browser request passes a URL with one of the headers defined, then Oracle Web Cache serves the document from its cache.

To specify HTTP request headers for multiple-version documents, select one of the headers in the **Multiple Documents with Same URL by Other Headers** column of the Edit/Create Cacheability Rule dialog box.

See Also: ["Configuring Cacheability Rules"](#) on page 6-5

Configuring Rules for Personalized Pages

You can specify cacheability rules for personalized pages that use personalized attributes or session-encoded URLs.

Personalized attributes are often in the form of "Hello, <Name>" or "Name". Personalized attributes can come in other forms, such as icons, addresses, or shopping cart snippets. You can configure Oracle Web Cache to cache the instructions for substituting values for personalized attributes based on the information contained within a cookie or the embedded URL parameter.

Session-encoded URLs enable Web sites to keep track of user sessions through session information contained within HTML tags. Oracle Web Cache can cache the instructions for replacing session information for one user with another based on the personal information contained within a cookie or as an embedded URL parameter.

See Also:

- ["Personalized Attributes"](#) on page 2-9 for an overview and an example scenario
- ["Session-Encoded URLs"](#) on page 2-12 for an overview and an example scenario

To create rules for personalized pages:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager"](#) on page 4-2

2. As necessary, create attribute definition(s) for those pages with personalized attributes and session definition(s) for those pages with session-encoded URLs:
 - a. In the navigator pane, select **Administering Web Sites > Session Management > Session/Personalized Attribute Definitions**.
The Session/Personalized Attribute Definitions page appears in the right pane.
 - b. In the Session/Personalized Attribute Definitions page, choose **Add**.
The Edit/Create Session/Personalized Attribute Definition dialog box appears.
 - c. In the **Session Name** field, enter an easy-to-remember unique name for the attribute or session.

For example, if the attribute is for a personalized greeting that uses the first name, you could enter `first_name01` for the session name.

- d. Enter the cookie name in the **Cookie Name** field and/or the embedded URL parameter in the **URL Parameter** field.

If you enter both a cookie name and an embedded URL parameter, keep in mind that both must support the same personalized attribute or session substitution. If they support different substitutions, create separate personalized definitions. You can specify up to 20 definitions for each page.

Note: Ensure that the size of cookies is not greater than 3KB.

- e. Choose **Submit**.

- f. Configure the pages that use personalized attributes with the `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` as follows:

```
<!-- WEBCACHETAG="personalized_attribute"-->
personalized attribute HTML segment
<!-- WEBCACHEEND-->
```

The personalized attribute information can be any valid HTML segment with beginning (`<tag>`) and ending (`</tag>`) HTML tags.

Ensure both tags have a space after `<!--`.

3. Create a cacheability rule for the personalized pages, as described in ["Configuring Cacheability Rules"](#) on page 6-5. In Step 6 of the procedure, select **Yes** in the **Personalized Pages** row of the Edit/Create Cacheability Rule dialog box, and then select one of the following options:

- **pages do not contain HREFs that are session encoded URLs** to cache substitution instructions for only personalized attributes
- **pages contain HREFs that are session encoded URLs** to cache substitution instructions for both personalized attributes and session-encoded URLs

Note: If a request comes in without the cookie or embedded URL parameter, Oracle Web Cache substitutes the personalized attribute or session ID information with an empty string. In addition, for session-encoded URLs, session establishment is delayed until there is a cache miss, as described in "[Session-Encoded URLs](#)" on page 2-12.

If you want to instead require that the request get the cookie or embedded URL parameter settings from the application Web server, perform these steps:

1. Create a session-related caching rule for the page to track the session, as described in "[Configuring Rules for Pages with Session Tracking](#)" on page 6-24.

2. In Step 6 of the procedure, select **YES** as the response.

3. In Step 7 of the procedure, select **NO** as the response.

4. Apply changes and restart Oracle Web Cache:

- a. In the Oracle Web Cache Manager main window, choose **Apply Changes**.
- b. In the navigator pane, select **Administering Oracle Web Cache > Web Cache Operations**.

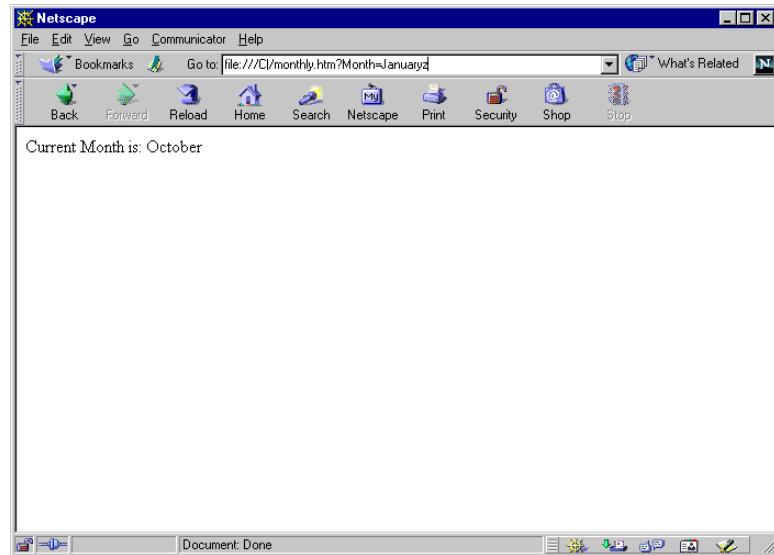
The Oracle Web Cache Operations page appears in the right pane.

- c. In the Oracle Web Cache Operations page, choose **Stop** and then **Start** to restart Oracle Web Cache.

Example: Personalized Page Configuration

To understand how to cache personalized content, consider the HTML page `monthly.htm` in **Figure 6–2**.

Figure 6–2 monthly.htm



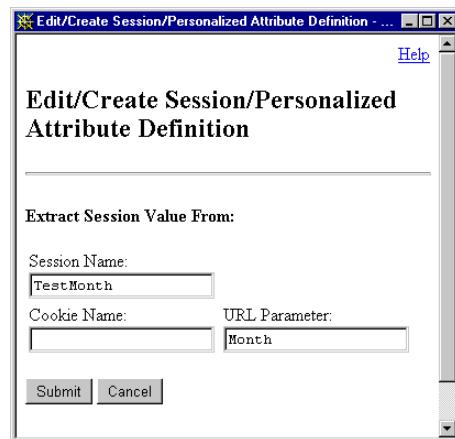
October is personalized content that can be substituted with other values.

The page has a URL of `monthly.htm?Month=month`, where Month is an embedded URL parameter.

The following steps were performed to cache `monthly.htm` and its personalized content.

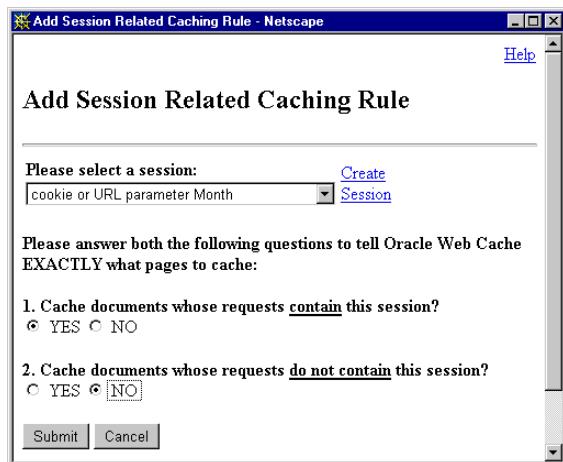
1. A personalized attribute of `TestMonth` was mapped to the embedded URL parameter `Month` in the Edit/Create Session/Personalized Attribute Definition dialog box.

Figure 6–3 Edit/Create Session/Personalized Attribute Definition Dialog Box



2. A session-related caching rule was created that uses the embedded URL parameter Month in the Add Session Related Caching Rule dialog box.

Figure 6-4 Add Session Related Caching Rule Dialog Box



See Also: "Configuring Rules for Pages with Session Tracking" on page 6-24 for more information about creating session-related caching rules

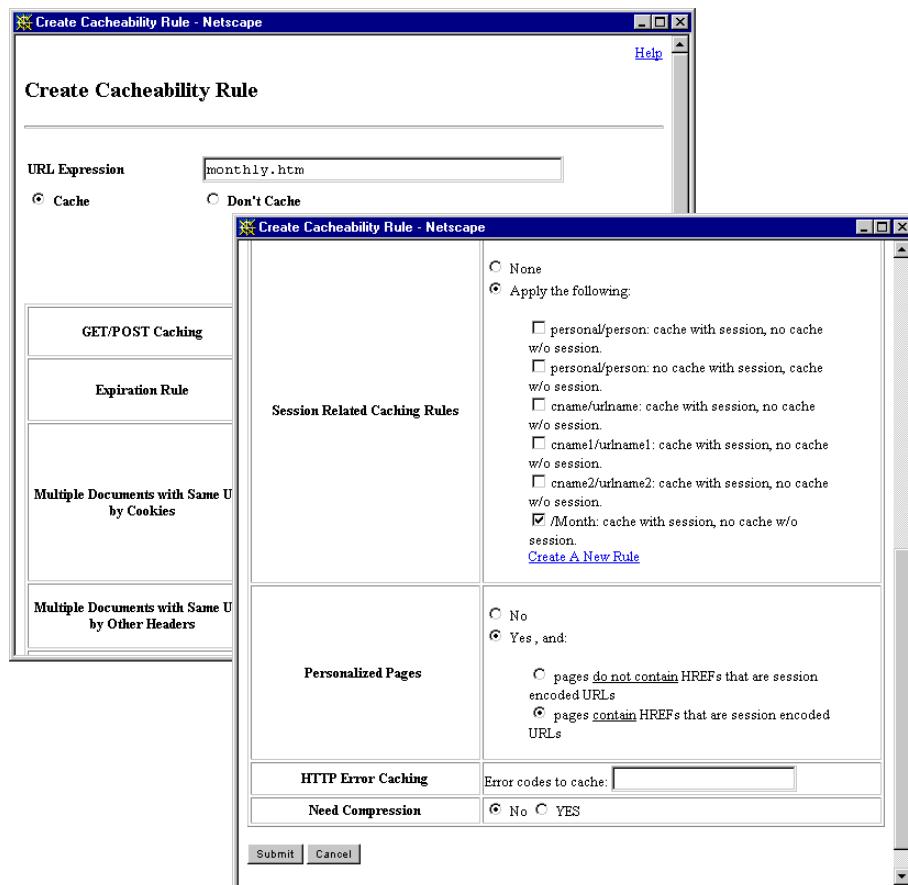
3. The <!-- WEBCACHETAG> and <!-- WEBCACHEEND--> HTML tags were added to monthly.htm.

Current Month is:

```
<!-- WEBCACHETAG="TestMonth"-->October<!-- WEBCACHEEND-->
```

4. A cacheability rule is created for `monthly.htm` in the Create Cacheability Rules dialog box
 - a. In the **Session-Related Caching Rule** row, the session-related caching rule for the embedded URL Month was chosen.
 - b. In the **Personalized Pages** row, **Yes** and **pages contain HREFs that are session encoded URLs** are chosen to cache substitution instructions for both personalized attributes and session-encoded URLs.

Figure 6–5 Create Cacheability Rule Dialog Box



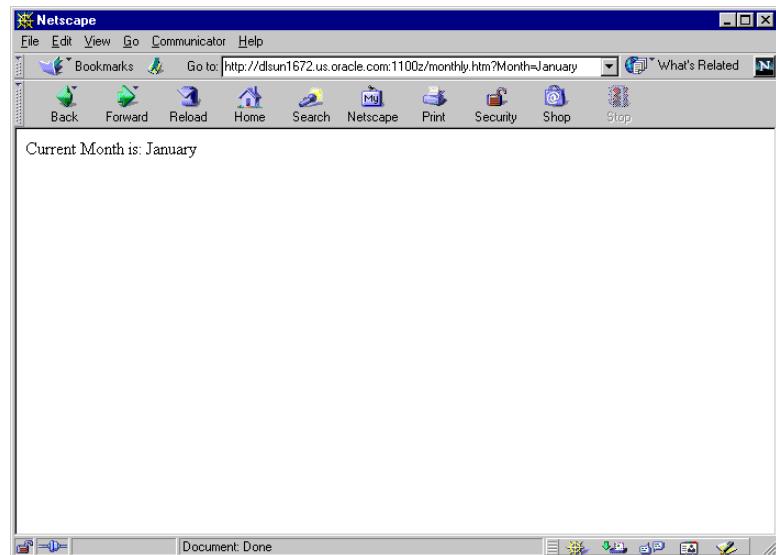
5. The configuration changes are applied:
 - a. In the Oracle Web Cache Manager main window, **Apply Changes** is chosen.
 - b. In the Oracle Web Cache Operations page, Oracle Web Cache is stopped and then restarted.

To verify that Oracle Web Cache was caching `monthly.htm`, requests for `monthly.htm` were sent to Oracle Web Cache:

1. An initial request for `monthly.htm` at URL `monthly.htm?Month=October` was requested. Because the initial request was forwarded by Oracle Web Cache to the application Web server, the value `October` was required for the `Month` parameter. This initial request inserted `monthly.htm` into the cache.
2. A subsequent request for `monthly.htm` was sent to URL `monthly.htm?Month=January`.

Oracle Web Cache substituted `October` with the value of `January`.

Figure 6–6 monthly.htm When Cached



Configuring Rules for Pages with Session Tracking

See Also: ["Session Tracking"](#) on page 2-11 for an overview

You can configure cacheability rules for pages that use session ID information, enabling Oracle Web Cache to serve the same page for multiple user sessions.

Here's how caching of session tracking pages works: When a user first accesses a Web site that uses session IDs, the application Web server assigns a unique session ID to the user. Session IDs are contained within a cookie or embedded in the URL as a parameter. If you configure Oracle Web Cache to cache the pages that use a session ID, subsequent requests that pass the cookie or embedded URL parameter are served from the cache.

Note that Oracle Web Cache ignores the values of session cookies. The response from the application Web server is cached, even if the response session cookie value does not match the request session cookie value. If you do not want the response cached when there is a value mismatch, then modify the application to instead send a non-200 status code as the response.

To create caching rules for pages that support session tracking:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager"](#) on page 4-2

2. In the navigator pane, select **Administering Web Sites > Cacheability Rules > Session-Related Caching Rules**.

The Session Related Caching Rules page appears in the right pane.

3. In the Session Related Caching Rules page, choose **Create or Add**.

The Add Session Related Caching Rule dialog box appears.

4. From the **Please select a session** list, select a session and skip to Step 6.

If the sessions listed do not contain the definition you require, choose **Create Session** to create a new session definition. The Edit/Create Session/Personalized Attribute Definition dialog box appears. Continue to Step 5.

5. Create a session definition in the Edit/Create Session/Personalized Attribute Definition dialog box:
 - a. In the **Session Name** field, enter an easy-to-remember unique name for the session.
 - b. Enter the cookie name in the **Cookie Name** field and/or the embedded URL parameter in the **URL Parameter** field.

If you enter both a cookie name and an embedded URL parameter, keep in mind that both must be used to support the same session. If they support different sessions, create separate session definitions. You can specify up to 20 definitions for each page.

Note: When a session cookie expires, the browser removes the cookie and subsequent requests for the document are directed to the application Web server. To avoid pages being served past the browser session expiration time, ensure that the session cookie expires before the application Web server expires the browser session.

- c. Choose **Submit**.
6. For the **Cache documents whose requests contain this session?** prompt in the Add Session Related Caching Rule dialog box, select either YES or NO:
 - Select **YES** to cache versions of documents that use the session information associated with the selected session.
 - Select **NO** to not cache versions of documents that use the session information.
7. For the **Cache documents whose requests do not contain this session?** prompt, choose either YES or NO:
 - Select **YES** to cache versions of documents that do not use the session information. This enables Oracle Web Cache to serve documents from the cache for Web browser requests without the session information.
 - Select **NO** to not cache versions of documents that do not use the session information.

Note: If you select **YES** in both Steps 6 and 7, two different versions of the document are cached. Oracle Web Cache serves one version to those requests that support session cookie or parameter and serves the other version to those requests that do not support the session cookie or parameter.

Note: If a request for a page that supports personalized attributes or session-encoded URLs comes in without the cookie or embedded URL parameter, Oracle Web Cache substitutes the personalized attribute or session ID information with an empty string. In addition, for session-encoded URLs, session establishment is delayed until there is a cache miss, as described in ["Session-Encoded URLs" on page 2-12](#).

If you want to instead require that the request get the cookie or embedded URL parameter settings from the application Web server, select **YES** in Step 6 and **NO** in Step 7.

8. Choose **Change URL Association** to associate the rule with a URL.
The Change Policy-URL Association dialog box appears.
9. Select a URL from the right list, and then choose the **Make Association** button.
The selected URL moves to the left list.
10. In the Add Session Related Caching Rule dialog box, choose **Submit**.
11. Repeat Steps 3 through 10 for each rule.
12. Apply changes and restart Oracle Web Cache:
 - a. In the Oracle Web Cache Manager main window, choose **Apply Changes**.
 - b. In the navigator pane, select **Administering Oracle Web Cache > Web Cache Operations**.
The Oracle Web Cache Operations page appears in the right pane.
 - c. In the Oracle Web Cache Operations page, choose **Stop** and then **Start** to restart Oracle Web Cache.

Tip: For some pages like the home page, you may want the first request to be served from the cache rather than from the application Web server. You can achieve this by following these steps:

1. Create a blank page that redirects to the home page.
2. Move the session creation from the home page to the blank page.
3. Create a session-related caching rule for the blank page to track the session.
4. Create a cacheability rule for the home page, as described in ["Configuring Cacheability Rules"](#) on page 6-5.

Configuration Considerations for Web Sites with Multiple Application Web Servers

This chapter describes additional configuration options available for deployments with two or more application Web servers. This chapter contains these topics:

- [Configuring Load Balancing](#)
- [Binding a Session to an Application Web Server](#)

Configuring Load Balancing

For those requests that Oracle Web Cache cannot serve, you can distribute the requests over a set of application Web servers with Oracle Web Cache's **load balancing** feature. To configure load balancing, you prescribe the relative load of each application Web server.

See Also: ["Load Balancing of Application Web Servers"](#) on page 1-9 for an overview of load balancing

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager"](#) on page 4-2

2. Configure application Web server settings in the Edit/Create Application Web Server dialog box, following the steps in ["Task 4: Specify Web Site Settings"](#) on page 5-7. In Step 2 of the procedure, enter the number of concurrent connections that the application Web server can sustain in the **Capacity** field. The weighted load percentage of each application Web server is derived from the entered capacity.

When load balancing is configured and an application Web server is no longer available, Oracle Web Cache automatically performs backend **failover** of the application Web servers. Oracle Web Cache knows if an application Web server is down when there are five continuous request failures to the server. An application Web server can become unavailable if it is taken down for reconfiguration or there is a network or hardware failure. In these scenarios, Oracle Web Cache automatically distributes the load over the remaining application Web servers and polls the failed application Web server for its current up/down status every 60 seconds until it is back online. Existing requests to the failed application Web server result in errors. However, new requests are directed to the other application Web servers. When the failed server returns to operation, Oracle Web Cache includes it in the load distribution.

Binding a Session to an Application Web Server

See Also: ["Application Web Server Binding" on page 1-12](#) for an overview of application Web server binding

You can configure Oracle Web Cache to support application Web server **session binding**, whereby a user session is bound to an application Web server in order to maintain state for a period of time. To utilize this feature, the application Web server itself must maintain state, that is, it must be stateful.

As long as the session information is contained within a **session cookie** or an embedded URL parameter, Oracle Web Cache can keep track of sessions between Web browsers and application Web servers. This enables Oracle Web Cache to bind a particular user session to a specific application Web server.

To configure Oracle Web Cache to support binding a user session to application Web servers that are stateful:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager" on page 4-2](#)

2. In the navigator pane, select **Administering Web Sites > Session Management > Session Binding**.

The Application Web Server Session Binding page appears in the right pane.

3. In the Application Web Server Session Binding page, choose **Edit**.

The Change Application Web Server Session Binding dialog box appears.

4. From the **Please select a session** list, select a session and skip to Step 7.

If the sessions listed do not contain the definition you require, choose **Create A New Session Definition** to create a new session definition. The Edit/Create Session Definitions dialog box appears. Continue to Step 5.

5. Create a session definition in the Edit/Create Session Definitions dialog box:

- a. In the **Session Name** field, enter an easy-to-remember unique name for the attribute.
- b. Enter the cookie name in the **Cookie Name** field and/or the embedded URL parameter in the **URL Parameter** field.

If you enter both a cookie name and an embedded URL parameter, keep in mind that both must be used to support the same session. If they support

different sessions, create separate session definitions. You can specify up to 20 session definitions for each page.

Note: When a session cookie expires, Oracle Web Cache does not continue to bind the user session to the application Web server. Instead, Oracle Web Cache uses load balancing to choose an application Web server. To avoid pages being served past the browser session expiration time, ensure that the session cookie expires before the application Web server expires the browser session.

- c. Choose **Submit**.
6. In the Change Application Web Server Session Binding dialog box, select the session definition from the **Please select a session** list.
7. In the **Inactivity Timeout** field, enter the number of minutes you want Oracle Web Cache to wait before timing out an inactive session to the application Web server. Oracle Corporation recommends setting the value to a higher value than the inactivity timeout set for the Web site.
8. Choose **Submit**.
9. Apply changes and restart Oracle Web Cache:
 - a. In the Oracle Web Cache main window, choose **Apply Changes**.
 - b. In the navigator pane, select **Administering Oracle Web Cache > Web Cache Operations**.
The Oracle Web Cache Operations page appears in the right pane.
- c. In the Oracle Web Cache Operations page, choose **Stop** and then **Start** to restart Oracle Web Cache.

Administering Oracle Web Cache

This chapter explains how to perform administrative tasks to Oracle Web Cache. It contains these topics:

- [Starting and Stopping Oracle Web Cache](#)
- [Invalidating Documents in the Cache](#)
- [Evaluating Event Logs](#)
- [Evaluating Access Logs](#)

Starting and Stopping Oracle Web Cache

Anytime Oracle Web Cache's configuration is modified, Oracle Web Cache must be stopped and restarted. To start and stop Oracle Web Cache, use either Oracle Web Cache Manager or the `webcachectl` utility.

When you start Oracle Web Cache from the `webcachectl` utility, the admin server process for the administrative interface and the cache server process for the actual cache are started. The Oracle Web Cache Manager starts only the cache server process. To initialize Oracle Web Cache for the first time, use the `webcachectl` utility to start both processes.

Use Oracle Web Cache Manager...	Use the <code>webcachectl</code> Utility...
To start the cache server process:	To start Oracle Web Cache:
<ol style="list-style-type: none">1. Start Oracle Web Cache Manager. See Also: "Starting Oracle Web Cache Manager" on page 4-22. In the navigator pane, select Administering Oracle Web Cache > Web Cache Operations. The Oracle Web Cache Operations page appears in the right pane.3. In the Oracle Web Cache Operations page, choose Start or Stop.	<ol style="list-style-type: none">1. Determine the status of Oracle Web Cache. From the command line, enter: <code>webcachectl status</code> If the following message appears, Oracle Web Cache is not running. Continue to Step 2. Oracle Web Cache admin server is NOT running. Oracle Web Cache cache server is NOT running. If the following message appears, Oracle Web Cache is already running. Oracle Web Cache admin server is running (pid=pid). Oracle Web Cache cache server is running (pid=pid)
	<ol style="list-style-type: none">2. Start Oracle Web Cache. From the command line, enter: <code>webcachectl start</code> The following message appears: Oracle Web Cache started To stop Oracle Web Cache, from the command line, enter: <code>webcachectl stop</code> The following message appears: Oracle Web Cache admin server stopping. Oracle Web Cache cache server stopping.

On Windows NT, Oracle Web Cache can also be started through the Control Panel:

1. Double-click the Services icon in the Control Panel window.
2. Select the Oracle*HOME_NAME*WebCacheAdmin service to start the `admin` server.
3. Choose Start to start the service.
4. Select the Oracle*HOME_NAME*WebCache service to start the `cache` server.
5. In the Services window, choose Close.

Invalidating Documents in the Cache

Invalidation messages are sent to Oracle Web Cache to an invalidation listening port through HTTP POST messages. The invalidation messages identify the documents to be invalidated.

This section contains the following invalidation-related topics:

- [Setting the Invalidation Port Number](#)
- [Sending Invalidation Messages](#)
- [Invalidation Examples](#)

Setting the Invalidation Port Number

By default, Oracle Web Cache listens for invalidation requests at port 4001.

To change the default port number:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager" on page 4-2](#)

2. Change the port numbers:

- a. In the navigator pane, select **Administering Oracle Web Cache > Oracle Web Cache Invalidation/Statistics Port**.

The Oracle Web Cache Invalidation/Statistics Port page appears in the right pane.

- b. In the Oracle Web Cache Invalidation/Statistics Port page, choose **Edit**.

The Change Invalidation/Statistics Port dialog box appears.

- c. In the **New Invalidation Port** field, enter the new port.

- d. Choose **Submit**.

3. Apply changes and restart Oracle Web Cache:

- a. In the Oracle Web Cache Manager main window, choose **Apply Changes**.

- b. In the navigator pane, select **Administering Oracle Web Cache > Web Cache Operations**.

The Oracle Web Cache Operations page appears in the right pane.

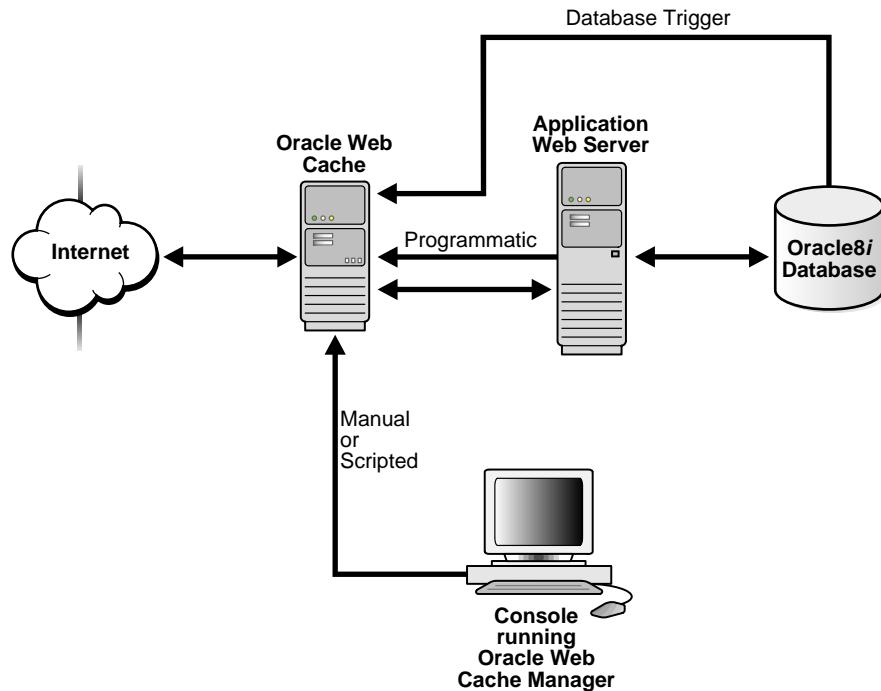
- c. In the Oracle Web Cache Operations page, choose **Stop** and then **Start** to restart Oracle Web Cache.

Sending Invalidation Messages

Invalidation messages are HTTP POST message written in **Extensible Markup Language (XML)** syntax. The contents of the XML message body tells the cache which URLs to mark as invalid. As shown in [Figure 8-1](#), invalidation messages can be sent using one of the following methods:

- Manually, using either Oracle Web Cache Manager or telnet
- Automatically, using database triggers, scripts, or applications

Figure 8-1 Invalidation



This section describes how to send invalidation messages using one of the following methods:

- [Manual Invalidation Using Telnet](#)
- [Manual Invalidation Using Oracle Web Cache Manager](#)
- [Automatic Invalidation Using Database Triggers](#)
- [Automatic Invalidation Using Scripts](#)
- [Automatic Invalidation Using Applications](#)

Manual Invalidation Using Telnet

When you send an invalidation message with a HTTP POST message, you specify the host name of Oracle Web Cache, the invalidation listening port number, and the invalidation message.

For example, if you were using telnet, you would send an invalidation message using the following procedure:

1. Connect to Oracle Web Cache at the invalidation listening port:

```
telnet webcache_host invalidation_port
```

2. Once you have telneted to the port, specify a POST message header and authenticate the user invalidator using Base64 encoding string with the following syntax.

```
POST /x-oracle-cache-invalidate http/1.0|1
Authorization: BASIC <base64 encryption of invalidator:invalidator_password>
content-length:#bytes
```

An example of Authorization: BASIC <base64 encryption of invalidator:invalidator_password> follows:

```
Authorization: BASIC aW52YWxpZGF0b3I6YWRtaW4=
```

In this example, aW52YWxpZGF0b3I6YWRtaW4= is "invalidator:admin" encoded.

See Also: ["Task 2: Modify Security Settings" on page 5-2](#) for further information about changing the invalidation password

3. Enter one carriage return.

4. Send the invalidation message with the following XML syntax:

```
<?xml version="1.0"?>

<!DOCTYPE INVALIDATION SYSTEM "internal:///invalidation.dtd">
<INVALIDATION>
  <URL EXP="URL" PREFIX="YES|NO">
    <VALIDITY LEVEL="validity" REFRESHTIME="seconds"/>
    <COOKIE NAME="cookie_name" VALUE="value" NONEXIST="YES|NO"/>
    <HEADER NAME="HTTP_request_header" VALUE="value" />
  </URL>
</INVALIDATION>
```

Table 8-1 Invalidation Message Syntax

Invalidation Element/Attribute	Description
URL	Required element in the invalidation message. You can specify more than one URL element in the message.
EXP	Required attribute for the URL element. Specify the URL of the documents you want to invalidate.
PREFIX	Optional attribute for the URL element. Specify YES to invalidate documents that match a prefix-based regular expression . YES limits the URL expression (EXP) to a limited form of regular expression, whereby the URL path (from first "/" to the last "/") matches the prefix of the URL. For example, if you have the URL expression set to /cec/cstage\?ecaction=viewitem, then /cec/ is the common prefix of the URLs to invalidate. Therefore, URLs such as /cec/cstage?ecaction=viewitem&zip=94405 and /cec/cstage?ecaction=viewitem&zip=94305 match, but /usa/cec/cstage?ecaction=viewitem&zip=94209 does not match. The path section cannot contain any reserved regular expression characters. Reserved regular expression characters include periods (.), question marks (?), asterisks (*), brackets ([]), curly braces ({ }), carats (^), dollar signs (\$), or backslashes (\). However, the part after the path, which is usually the file name, can include regular expression characters. Specify NO to only invalidate documents contained within the URL. NO also means that the URL expression (EXP) is interpreted literally.

Invalidation Element/Attribute	Description
	EXAMPLE 1
	Validation Intent: Invalidate sample.gif contained within the /cec/cstage/graphic*/sample.gif directory.
	URL Expression Entered: /cec/cstage/graphic*/sample.gif
	Result:
	If PREFIX is set to NO, then /cec/cstage/graphic*/sample.gif matches and Oracle Web Cache invalidates only /cec/cstage/graphic*/sample.gif. The "*" and "." characters are not interpreted as regular expression characters.
	If PREFIX is set to YES, then /cec/cstage/graphic*/sample.gif is not invalidated. This is because the path includes "*", a reserved regular expression character that cannot be included in the path section, and the "." in the file name is interpreted as a regular expression character to match any character. If "*" is replaced with the exact directory name and the "." is escaped, as in /cec/cstage/graphicsales/sample\.gif, then sample.gif is invalidated.
	EXAMPLE 2
	Validation Intent: Invalidate documents contained in the /cec/cstage?ecaction=ecsectview directory.
	URL Expression Entered: /cec/cstage?ecaction=ecsectview
	Result:
	If PREFIX is set to NO, then /cec/cstage?ecaction=ecsectview matches and Oracle Web Cache invalidates only /cec/cstage?ecaction=ecsectview. The "?" character is not interpreted as a regular expression character.
	If PREFIX is set to YES, then /cec/cstage?ecaction=ecsectview is not invalidated. This is because the "?" in the file name is interpreted as a regular expression character to match zero or one occurrence of the character "e" it follows. If "?" is escaped, as in /cec/cstage\?ecaction=ecsectview, then all documents within the prefix /cec/cstage?ecaction=ecsectview are matched and invalidated.

Invalidation Element/Attribute	Description
VALIDITY	Required element in the invalidation message.
LEVEL	<p>Optional attribute for the VALIDITY element.</p> <p>Specify the validity level of the documents. The validity level ranges from 0 (for the least valid) to 9 (for the most valid).</p> <p>The higher the validity level, the longer Oracle Web Cache serves documents stale from the cache before removing them. Oracle Web Cache serves documents with a low validity level for a short amount of time before removing them. After documents are removed, Oracle Web Cache retrieves new versions of the documents from the application Web servers.</p> <p>A validity level of 0 means Oracle Web Cache will remove documents immediately. The default validity level is 0.</p>
REFRESHTIME	<p>Optional attribute for the VALIDITY element.</p> <p>Specify the maximum time that documents can reside in the cache.</p>
COOKIE	<p>Optional element in the invalidation message.</p>
NAME	<p>Required attribute for the COOKIE element.</p> <p>Specify the cookie name to invalidate documents based on the cookie. The name must match the cookie name of a multiple-version cacheability rule associated with this URL.</p>
VALUE	<p>Optional attribute for the COOKIE element.</p> <p>Specify the value of the cookie.</p>
NONEEXIST	<p>Optional attribute for the COOKIE element.</p> <p>Specify YES to invalidate documents for requests without this cookie; specify NO to invalidate documents for requests with this cookie. NO is the default.</p>
HEADER	<p>Optional element in the invalidation message.</p>
NAME	<p>Required attribute for the HEADER element.</p> <p>Specify the HTTP request header and its value to invalidate based on the request header. The name must match the header of a multiple-version cacheability rule associated with this URL.</p>
VALUE	<p>Optional attribute for the HEADER element.</p> <p>Specify the value of the header.</p>

See Also: "Invalidation Request DTD" on page C-2 for further information about invalidation request syntax

Invalidation responses are returned in the following format:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT [
<!ELEMENT INVALIDATIONRESULT (URL+)>
<!ELEMENT URL   EMPTY>
<!ATTLIST URL
      EXPR      CDATA  #REQUIRED
      ID        CDATA  #REQUIRED
      STATUS    CDATA  #REQUIRED
      NUMINV   CDATA  #REQUIRED
>
]>
<INVALIDATIONRESULT>
<URL EXPR="URL" ID="ID" STATUS="status" NUMINV="number">
</INVALIDATIONRESULT>
```

Table 8–2 *Invalidation Response Syntax*

Invalidation Element/Attribute	Description
URL	
EXP	Specifies the URL of the document(s) that you requested to be invalidated
ID	Sequence number of all the URLs sent in the invalidation response. If there are multiple URLs specified in the invalidation message, then the sequence number starts at 1 for the first URL and sequentially numbers for each additional URL.
STATUS	Status of the invalidation. Status can be one of the following: <ul style="list-style-type: none">■ SUCCESS for successful invalidations■ URI NOT CACHEABLE for documents not in the cache■ URI NOT FOUND for documents not found
NUMINV	Number of documents invalidated

See Also: ["Invalidation Response DTD"](#) on page C-5 for further information about invalidation response syntax

Manual Invalidation Using Oracle Web Cache Manager

Oracle Web Cache Manager provides an easy-to-use interface for invalidating cached objects. Under the covers, the message mechanics are much like the telnet example. The advantage of using Oracle Web Cache Manager is that the administrator is isolated from the intricacies of the HTTP and XML formats, and consequently, there is less chance for error. The administrator need only specify which objects to invalidate and how invalid those objects should be.

To invalidate documents with Oracle Web Cache Manager:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager"](#) on page 4-2

2. In the navigator pane, select **Administering Oracle Web Cache > Cache Cleanup**.

The Cache Cleanup page appears in the right pane.

3. In the **URL Scope** section, specify which documents you want to invalidate:

ALL	Select to remove all documents from the cache.
URL or URL Prefix	Enter the URL of the document(s) contained within a specific URL or URL prefix pattern.
apply to all documents matching prefix	Select to invalidate documents that match a prefix-based regular expression. This option limits the URL to a limited form of regular expression, whereby the URL path (from first "/" to the last "/") matches the prefix of the URL. For example, if you have the URL set to /cec/cstage\?ecaction=viewitem, then /cec/ is the common prefix of the URLs to invalidate. Therefore, URLs such as /cec/cstage?ecaction=viewitem&zip=94405 and /cec/cstage?ecaction=viewitem&zip=94305 match, but /usa/cec/cstage?ecaction=viewitem&zip=94209 does not match. The path section cannot contain any reserved regular expression characters. Reserved regular expression characters include periods (.), question marks (?), asterisks (*), brackets ([]), curly braces ({ }), carats (^), dollar signs (\$), or backslashes (\). However, the part after the path, which is usually the file name, can include regular expression characters.
	Deselect to only invalidate documents contained within the URL. This also means that the URL expression is interpreted literally.

EXAMPLE 1

Invalidation Intent: Invalidate sample.gif contained within the /cec/cstage/graphic*/sample.gif directory.

URL Expression Entered:

/cec/cstage/graphic*/sample.gif

Result:

If this option is not selected, then

/cec/cstage/graphic*/sample.gif matches and Oracle Web Cache invalidates only

/cec/cstage/graphic*/sample.gif. The "*" and "." characters are not interpreted as regular expression characters.

If this option is selected, then

/cec/cstage/graphic*/sample.gif is not invalidated.

This is because the path includes "*", a reserved regular expression character that cannot be included in the path section, and the "." in the file name is interpreted as a regular expression character to match any character. If "*" is replaced with the exact directory name and the "." is escaped, as in /cec/cstage/graphicsales/sample\.gif, then sample.gif is invalidated.

EXAMPLE 2

Invalidation Intent: Invalidate documents contained in the /cec/cstage?ecaction=ecsectview directory.

URL Expression Entered:

/cec/cstage?ecaction=ecsectview

Result:

If this option is not selected, then

/cec/cstage?ecaction=ecsectview will match and invalidate only /cec/cstage?ecaction=ecsectview. The "?" character is not interpreted as a regular expression character.

If this option is selected, then

/cec/cstage?ecaction=ecsectview is not invalidated. This is because the "?" in the file name is interpreted as a regular expression character to match zero or one occurrence of the character "e" it follows. If "?" is escaped, as in /cec/cstage\?ecaction=ecsectview, then all documents within the prefix

/cec/cstage?ecaction=ecsectview are matched and invalidated.

4. In the **Action** section, specify how you want Oracle Web Cache to process invalid documents.

Remove immediately

Select this option to have Oracle Web Cache mark documents as invalid and then remove them immediately. A document is refreshed from the application Web server when the cache receives the next request for it.

Note: This is equivalent to a validity level of 0.

Refresh on demand as application Web server capacity permits AND no later than <time> after submission

Select this option to have Oracle Web Cache mark documents as invalid and then refresh them based on application Web server capacity. Enter the maximum time in which the documents can reside in the cache.

Optionally, select a validity level for the documents after they expire. Validity determines how long Oracle Web Cache will serve documents stale from the cache before marking them as invalid.

The higher the validity level, the longer Oracle Web Cache serves documents stale from the cache before removing them. Oracle Web Cache serves documents with a low validity level for a short amount of time before removing them. After documents are removed, Oracle Web Cache retrieves new versions of the documents from the application Web servers.

Note: Performance assurance heuristics apply when you configure documents to be refreshed based on when the application Web servers can refresh them; performance assurance heuristics do not apply when documents are immediately removed.

5. Choose **Submit**.

Automatic Invalidation Using Database Triggers

Database triggers are procedures that are stored in the database and activated ("fired") when specific conditions occur, such as adding a row to a table. You can use triggers to send invalidation messages. To this do, use the UTL_TCP Oracle supplied package to send invalidation messages through database triggers.

See Also:

- README.examples.txt in the \$ORACLE_HOME/webcache/examples directory on UNIX and *ORACLE_HOME\webcache\examples* directory on Windows NT for further information about using the `cre_invalid_trig.sql` script to create a database trigger and the `utl_proc.sql` script to demonstrate invalidation with database triggers
- *PL/SQL User's Guide and Reference*

Automatic Invalidation Using Scripts

Many Web sites use scripts for uploading new content to databases and file systems. A large online book retailer, for instance, might run a PERL script once a day in order to bulk load new book listings and price changes into its catalog database. The retailer would want the price changes and availability listings to be reflected in the item views and search results currently cached in Oracle Web Cache. To achieve this, the PERL script can be modified such that when the bulk loading operation has completed, the script will send an invalidation message to the cache invalidating all catalog views and search results. (Note that the invalidation message need not list every individual search page or item view that might be effected by the data change.) The performance assurance feature of Oracle Web Cache enables administrators to use broad brush strokes when invalidating content, making it safe to invalidate all catalog content even if only a fraction of that content has changed.

Automatic Invalidation Using Applications

Invalidation messages can also originate from a Web site's underlying application logic or from the content management application used to design Web pages. Oracle Web Cache ships with invalidation Java source that you can link with your applications for automatically generating invalidation messages.

See Also: README.examples.txt in the \$ORACLE_HOME/webcache/examples directory on UNIX and *ORACLE_HOME\webcache\examples* directory on Windows NT for further information about using the `Invalidate.java` file

Invalidation Examples

These examples require utilizing the POST method which also requires sending the number of bytes (or characters) in the content_length: #bytes portion of the header. Please note that one carriage return is required after the content_length: #bytes line and before the XML message or BODY information.

Example: Invalidating One Document

Invalidation message:

```
<?xml version="1.0" ?>

<!DOCTYPE INVALIDATION SYSTEM "internal:///invalidation.dtd">
<INVALIDATION>
  <URL EXP="/images/logo.gif" PREFIX="NO">
    <VALIDITY LEVEL="1"/>
  </URL>
</INVALIDATION>
```

Invalidation response:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT [
<!ELEMENT INVALIDATIONRESULT (URL+)>
<!ELEMENT URL   EMPTY>
<!ATTLIST URL
      EXPR      CDATA  #REQUIRED
      ID        CDATA  #REQUIRED
      STATUS    CDATA  #REQUIRED
      NUMINV   CDATA  #REQUIRED
>
]>
<INVALIDATIONRESULT>
<URL EXP="/images/logo.gif" ID="1" STATUS="SUCCESS" NUMINV="1">
</INVALIDATIONRESULT>
```

Example: InValidating a Subtree of Documents

InValidation message:

```
<?xml version="1.0"?>

<!DOCTYPE INVALIDATION SYSTEM "internal:///invalidation.dtd">
<INVALIDATION>
    <URL EXP="/images/" PREFIX="YES">
        <VALIDITY LEVEL="1" REFRESHTIME="60"/>
    </URL>
</INVALIDATION>
```

InValidation response:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT [
    <!ELEMENT INVALIDATIONRESULT (URL+)>
    <!ELEMENT URL EMPTY>
    <!ATTLIST URL
        EXPR      CDATA  #REQUIRED
        ID       CDATA  #REQUIRED
        STATUS    CDATA  #REQUIRED
        NUMINV   CDATA  #REQUIRED
    >
]>
<INVALIDATIONRESULT>
    <URL EXP="/images/" ID="1" STATUS="SUCCESS" NUMINV="125">
</INVALIDATIONRESULT>
```

Example: Invalidating All Documents for a Web Site

Invalidation message:

```
<?xml version="1.0"?>

<!DOCTYPE INVALIDATION SYSTEM "internal:///invalidation.dtd">
<INVALIDATION>
  <URL EXP="/" PREFIX="YES">
    <VALIDITY LEVEL="0"/>
  </URL>
</INVALIDATION>
```

Invalidation response:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT [
<!ELEMENT INVALIDATIONRESULT (URL+)>
<!ELEMENT URL EMPTY>
<!ATTLIST URL
          EXPR      CDATA  #REQUIRED
          ID        CDATA  #REQUIRED
          STATUS    CDATA  #REQUIRED
          NUMINV   CDATA  #REQUIRED
>
]>
<INVALIDATIONRESULT>
<URL EXP="/"
      ID="1"
      STATUS="SUCCESS"
      NUMINV="5347">
</INVALIDATIONRESULT>
```

Evaluating Event Logs

Oracle Web Cache events and errors are stored in an event log. The event log can help you determine what documents or objects have been inserted into the cache. It can also identify listening port conflicts or startup and shutdown issues. The event log has a file name of `event_log` and is stored in `$ORACLE_HOME/webcache/logs` on UNIX and `ORACLE_HOME\webcache\logs` on Windows NT.

See Also: [Appendix D, "Event Log Messages"](#) for descriptions of the most common event log messages

Format of the Event Log File

Events are formatted into the following fields:

Timestamp Information/Warning/Error Message

Event Log Examples

Example: Event Log with Startup Entries

[Figure 8-2](#) shows an event log excerpt with successful startup entries.

Figure 8-2 Event Log with Successful Startup Entries

```
19/Sep/2000:10:20:56 -0500 -- Information: Max Connect Count exceeds compile  
time limit - defaulting to  
connect limit: ( 854 )  
19/Sep/2000:10:20:57 -0500 -- Information: Listening on ADMINISTRATION port 5000  
address 0.0.0.0  
19/Sep/2000:10:20:57 -0500 -- Information: The admin server started successfully  
19/Sep/2000:10:20:57 -0500 -- Information: Max Connect Count exceeds compile  
time limit - defaulting to  
connect limit: ( 854 )  
19/Sep/2000:10:20:57 -0500 -- Information: Listening on NORM port 1100 address  
0.0.0.0  
19/Sep/2000:10:20:57 -0500 -- Information: Listening on INVALIDATION port 5001  
address 0.0.0.0  
19/Sep/2000:10:20:57 -0500 -- Information: Listening on STATISTICS port 5002  
address 0.0.0.0  
19/Sep/2000:10:20:58 -0500 -- Information: The cache server started successfully  
19/Sep/2000:10:20:58 -0500 -- Information: The cache server is started by the  
admin server at startup
```

Example: Event Log with Unsuccessful Startup Entries

[Figure 8–3](#) shows an event log excerpt with unsuccessful startup events. Oracle Web Cache is unable to listen on port 1100, because it is already in use. This can occur if Oracle Web Cache is already running and listening on that port or another application is using that port.

Figure 8–3 Event Log with an Unsuccessful Startup

```
14/Sep/2000:16:37:41 -0800 -- Error: A failure occurred ( Address already in use ) when assigning a port ( domain: <NONE>, address: 0.0.0.0, port: 1100 ). Change PORT attribute of the LISTEN element in the configuration file to a suitable unused port.  
14/Sep/2000:16:37:41 -0800 -- Error: Failed to start the server.  
14/Sep/2000:16:37:41 -0800 -- Error: The server could not initialize  
14/Sep/2000:16:37:41 -0800 -- Information: The server is exiting
```

Example: Event Log with an Invalidation Entry

[Figure 8–4](#) shows an event log excerpt with an event associated with an invalidation request for the removal of document personal.htm.

Figure 8–4 Event Log with an Invalidation Entry

```
30/Sep/2000:22:52:52 -0500 -- Information: <Invalidation>1 URLs with prefix personal.htm have been successfully invalidated.
```

Example: Event Log with an Invalidation Message Error

[Figure 8–5](#) shows an event log excerpt with an XML invalidation message error. In this example, Oracle Web Cache is unable to parse the message.

Figure 8–5 Event Log with an Invalidation Message Error

Example: Errors in the XML parsing. Note: The configuration files and invalidation files use XML files.
19/Sep/2000:10:55:26 -0500 -- Error: Invalidation XML Buffer cannot be parsed.
19/Sep/2000:10:55:26 -0500 -- Error: XML parsing error.

Example: Event Log with Shutdown Entries

[Figure 8–6](#) shows an event log excerpt with typical shutdown entries.

Figure 8–6 Event Log with Shutdown Entries

```
14/Sep/2000:11:16:55 -0700 -- Information: SIGTERM caught - program will shut  
down once all connections are complete.  
14/Sep/2000:11:16:55 -0800 -- Information: SIGTERM caught - program will shut  
down once all connections are complete.  
14/Sep/2000:11:16:55 -0700 -- Information: The server is exiting  
14/Sep/2000:11:16:55 -0800 -- Information: The server is exiting
```

Finding Errors in the Event Log

To list just the errors in the event log, use grep on UNIX. For example:

```
grep " Error:" error*
```

To list the errors by the current day, enter grep " Error:" event_log "dd/month/yyyy". For example:

```
grep " Error:" event_log | grep "19/Sep/2000"
```

To list errors by the current day and hour, enter grep " Error:" event_log "dd/month/yyyy:hh".

Configuring Event Logs

To establish event log configuration settings:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager"](#) on page 4-2

2. In the navigator pane, select **Administering Oracle Web Cache > Event Logging**.

The Event Logging page appears in the right pane.

3. In the Event Logging page, choose **Edit**

The Change Options for Event Logging dialog box appears.

4. In **Logging Time Format**, select either **Local** or **GMT** (Greenwich Mean Time) to modify the time stamp style associated with entries in the event log file.

Note: Oracle Corporation recommends using GMT whenever possible. Local can be CPU-intensive, because of the conversion process from GMT to Local time. This conversion process is supplied by the operation system. As such, Oracle Web Cache has no mechanism to improve the performance of the conversion process.

5. In **Verbose Logging**, select either **No** to log typical events or **Yes** to log typical events, plus application Web server events. Verbose event logs are used for debugging purposes. Therefore, select **Yes** if recommended by Oracle Support Services.
6. Choose **Submit**.
7. Apply changes and restart Oracle Web Cache:
 - a. In the Oracle Web Cache Manager main window, choose **Apply Changes**.
 - b. In the navigator pane, select **Administering Oracle Web Cache > Web Cache Operations**.
The Oracle Web Cache Operations page appears in the right pane.
 - c. In the Oracle Web Cache Operations page, choose **Stop** and then **Start** to restart Oracle Web Cache.

Evaluating Access Logs

Each Web site that Oracle Web Cache supports has its own access log. An access log contains information about the HTTP requests sent to Oracle Web Cache for a Web site. The access log has a file name of `access_log` and is stored by default in `$ORACLE_HOME/webcache/logs` on UNIX and `ORACLE_HOME\webcache\logs` on Windows NT. Note that Oracle Web Cache uses buffered logging for the access log, that is, it writes to the access log after the buffer is full.

Format of the Access Log File

You can configure the content of the access log file by defining the fields to appear for each HTTP request event. These fields are a part of the **ExtendedLogFile Format (XLF)**, which is a superset of the **CommonLogFile Format (CLF)**.

Table 8–3 lists the XLF fields you can select.

Table 8–3 XLF Fields for Access Logs

Field	Description
cs(User-Agent)	Information about the user agent originating the request
cs(Referer)	Allows the client to specify the address (URI) of the resource from which the Request-URI was obtained
c-auth-id	Username if the request contained an attempt to authenticate
bytes	Content-length of the transferred document
date	Date at which the transaction completed
time	Time at which the transaction completed
time-taken	Amount of time taken (in seconds) for transaction to complete
c-ip	Client's IP address and port
s-ip	Oracle Web Cache's IP address and port
sc-method	Oracle Web Cache-to-client HTTP request method (GET, POST, or others)
sc-status	Oracle Web Cache-to-client HTTP status code: <ul style="list-style-type: none"> ■ 1xx range messages are informational. ■ 2xx range messages indicate success. ■ 3xx range messages indicate redirection, indicating that further action must be taken in order to complete the request. ■ 4xx range messages indicate a client error. ■ 5xx range messages indicate a Oracle Web Cache error. See Also: http://www.ietf.org/rfc/rfc2616.txt for further information about HTTP status codes
cs-uri	Client-to-Oracle Web Cache URI

Table 8–3 XLF Fields for Access Logs

Field	Description
cs-uri-stem	Client-to-Oracle Web Cache stem portion of URI, omitting the query
cs-uri-query	Client-to-Oracle Web Cache query portion of URI, omitting the stem
<i>prefix(header)</i>	<i>header</i> is a HTTP header field and <i>prefix</i> is one of the following: c: Client s: Oracle Web Cache r: Remote cs: Client-to-Oracle Web Cache sc: Oracle Web Cache-to-client

If no fields are specified, then the following default CLF fields are used in the access log file:

```
c-ip cauth-id [clf-date] "request line" sc-status bytes
```

Access Log Examples

Figure 8–7 shows an example access log using the default CLF fields:

```
c-ip cauth-id [clf-date] "request line" sc-status bytes
```

The "request line" is represented by the "GET ...HTTP/1.0" portion of the request. The "request line" enables you to determine what is being accessed and the following *sc_status*, or HTTP status code, reports if the request was successfully completed.

Figure 8–7 Access Log

```
138.2.213.146 -- [19/Sep/2000:10:27:42 -0500] "GET /~ssandrew/personal.htm
HTTP/1.0" 200 2438
138.2.213.146 -- [19/Sep/2000:10:27:54 -0500] "GET
/~ssandrew/personal.htm?UserName=Bob HTTP/1.0"
200 2438
138.2.213.146 -- [19/Sep/2000:10:47:30 -0500] "GET /~ssandrew/count.sh
HTTP/1.0" 403 289
138.2.213.146 -- [19/Sep/2000:10:47:34 -0500] "GET /~ssandrew/sbin/count.sh
HTTP/1.0" 200 321
138.2.213.146 -- [19/Sep/2000:10:47:41 -0500] "GET /sbin/count.sh HTTP/1.0" 200
321
138.2.213.146 -- [19/Sep/2000:11:34:23 -0500] "GET /cache.htm HTTP/1.0" 200 250
138.2.213.146 -- [19/Sep/2000:11:38:23 -0500] "GET /cache.htm HTTP/1.0" 304 0
138.2.213.146 -- [19/Sep/2000:11:38:48 -0500] "GET /cache.htm HTTP/1.0" 304 0
206.223.27.37 -- [19/Sep/2000:15:14:29 -0500] "GET
```

```
~/ssandrew/personal.htm?UserName=Joe HTTP/1.0"
200 2438
206.223.27.37 - - [19/Sep/2000:15:17:12 -0500] "GET
~/ssandrew/personal.htm?UserName=Shehzaad
HTTP/1.0" 200 438
144.25.223.39 - - [19/Sep/2000:15:30:34 -0500] "GET /htdocs/coelist.html
HTTP/1.0" 200 4219
144.25.223.39 - - [19/Sep/2000:15:30:34 -0500] "GET /images/redheaderbanner.gif
HTTP/1.0" 200 1226
138.2.213.146 - - [19/Sep/2000:10:49:44 -0500] "GET /pls/coe/find_via_post
HTTP/1.0" 200 1119
138.2.213.146 - - [19/Sep/2000:10:49:44 -0500] "GET /ows-img/chalk.jpg HTTP/1.0"
404 284
130.35.35.21 - - [20/Sep/2000:00:36:35 -0500] "GET /images/support.jpg HTTP/1.0"
206 3106
130.35.35.21 - - [20/Sep/2000:00:36:35 -0500] "GET /images/ani_coe.gif HTTP/1.0"
206 73118
```

Example: Access Log with Reload Entries

Figure 8–8 shows an access log excerpt in which there are two Web browser reloads, followed by two shift reloads, and two more reloads.

Figure 8–8 Access Log with Reload Entries

```
138.2.213.146 - - [19/Sep/2000:11:04:24 -0500] "GET /cache.htm HTTP/1.0" 200 250
138.2.213.146 - - [19/Sep/2000:11:04:26 -0500] "GET /cache.htm HTTP/1.0" 200 250
138.2.213.146 - - [19/Sep/2000:11:29:24 -0500] "GET /cache.htm HTTP/1.0" 304 0
138.2.213.146 - - [19/Sep/2000:11:29:25 -0500] "GET /cache.htm HTTP/1.0" 304 0
138.2.213.146 - - [19/Sep/2000:11:29:30 -0500] "GET /cache.htm HTTP/1.0" 200 250
138.2.213.146 - - [19/Sep/2000:11:29:35 -0500] "GET /cache.htm HTTP/1.0" 200 250
```

Example: Access Log with Wrong Path Entry

[Figure 8–9](#) shows an access log excerpt in which a browser requested the wrong path. This is indicated by HTTP status code 403. The browser then requested the correct path. This is indicated by HTTP status code 200.

Figure 8–9 Access Log with Wrong Path Entry

```
38.2.213.146 -- [19/Sep/2000:10:47:30 -0500] "GET /~ssandrew/count.sh HTTP/1.0"
403 289
138.2.213.146 -- [19/Sep/2000:10:47:34 -0500] "GET /~ssandrew/sbin/count.sh
HTTP/1.0" 200 321
```

Example: Access Log with Status Code 404 Entry

[Figure 8–10](#) shows an access log excerpt in which a Oracle Web Cache cannot find any objects matching the requested URL /ows-img/chalk.jpg. This indicated by HTTP status code 404.

Figure 8–10 Access Log with HTTP Status Code 404 Entry

```
138.2.213.146 -- [19/Sep/2000:10:49:44 -0500] "GET /pls/coe/find_via_post
HTTP/1.0" 200 1119
138.2.213.146 -- [19/Sep/2000:10:49:44 -0500] "GET /ows-img/chalk.jpg HTTP/1.0"
404 284
```

Example: Access Log with Status Code 304 Entry

[Figure 8–11](#) shows an access log excerpt in which the first entry shows a Web browser request for /cache.htm being successfully completed. The second and third entries return a HTTP status code of 304, indicating that document has not been modified and does not need to be returned again.

Figure 8–11 Access Log with HTTP Status Code 304 Entry

```
138.2.213.146 -- [19/Sep/2000:11:34:23 -0500] "GET /cache.htm HTTP/1.0" 200 250
138.2.213.146 -- [19/Sep/2000:11:38:23 -0500] "GET /cache.htm HTTP/1.0" 304 0
138.2.213.146 -- [19/Sep/2000:11:38:48 -0500] "GET /cache.htm HTTP/1.0" 304 0
```

Configuring Access Logs

To enable access logging:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager" on page 4-2](#)

2. In the navigator pane, select **Administering Web Sites > Access Logging**.
The Access Logs page appears in the right pane.
3. In the Access Logs page, choose **Edit**
The Change Options for Access Logs dialog box appears.
4. In **Logging Enabled**, select **YES**.
5. In the **Logging Directory** field, enter the directory path where you want the log file written.
6. In **Logging Time Format**, select either **Local** or **GMT** (Greenwich Mean Time) to modify the time stamp style associated with entries in the access log file.

Note: Oracle Corporation recommends using GMT whenever possible. Local can be CPU-intensive, because of the conversion process from GMT to Local time. This conversion process is supplied by the operation system. As such, Oracle Web Cache has no mechanism to improve the performance of the conversion process.

7. From the **Rollover Frequency** list, select how often you want to change the frequency at which Oracle Web Cache will save current log information to `access_log.yyyymmdd` and write new log information to `access_log`. If you have a high-volume site, increase the frequency.
8. In the **XLF Fields** field, enter XLF fields to log. Separate fields by a space.
9. Choose **Submit**.
10. Apply changes and restart Oracle Web Cache:
 - a. In the Oracle Web Cache Manager main window, choose **Apply Changes**.
 - b. In the navigator pane, select **Administering Oracle Web Cache > Web Cache Operations**.

The Oracle Web Cache Operations page appears in the right pane.

- c. In the Oracle Web Cache Operations page, choose **Stop** and then **Start** to restart Oracle Web Cache.

To disable access logging:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager"](#) on page 4-2

2. In the navigator pane, select **Administering Web Sites > Access Logging**.
The Access Logs page appears in the right pane.
3. In the Access Logs page, choose **Edit**
The Change Options for Access Logs dialog box appears.
4. In **Logging Enabled**, select **NO**.
5. Choose **Submit**.
6. Apply changes and restart Oracle Web Cache:
 - a. In the Oracle Web Cache Manager main window, choose **Apply Changes**.
 - b. In the navigator pane, select **Administering Oracle Web Cache > Web Cache Operations**.
The Oracle Web Cache Operations page appears in the right pane.
- c. In the Oracle Web Cache Operations page, choose **Stop** and then **Start** to restart Oracle Web Cache.

Monitoring Performance

See Also: *Oracle Internet Application Server 8i Oracle HTTP Server powered by Apache Performance Guide* for TCP/IP performance tuning tips for the computer running Oracle Web Cache

This chapter describes how to gather performance statistics and how to interpret them. This chapter contains these topics:

- [Setting the Statistics Monitoring Port Number](#)
- [Monitoring Overall Cache Health](#)
- [Gathering Oracle Web Cache Performance Statistics](#)
- [Gathering Application Web Server Performance Statistics](#)

Setting the Statistics Monitoring Port Number

By default, Oracle Web Cache listens for statistics monitoring requests at port 4002.

To change the default port number:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager" on page 4-2](#)

2. Change the port numbers:

- a. In the navigator pane, select **Administering Oracle Web Cache > Invalidations/Statistics Port**.

The Oracle Web Cache Invalidations/Statistics Port page appears in the right pane.

- b. In the Oracle Web Cache Invalidations/Statistics Port page, choose **Edit**.

The Change Invalidations/Statistics Port dialog box appears.

- c. In the **Statistics Port** field, enter the new port.

- d. Choose **Submit**.

3. In the Oracle Web Cache main window, choose **Apply Changes**.

Monitoring Overall Cache Health

Oracle Web Cache provides a health monitor that enables you to quickly access overall cache performance.

To monitor overall cache health:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager" on page 4-2](#)

2. In the navigator pane, select **Administering Web Sites > Monitoring Oracle Web Cache > Health Monitor**.

The Oracle Web Cache Health Monitor page appears in the right pane.

[Table 9-1](#) describes the statistics for this page.

Table 9-1 Oracle Web Cache Health Monitor Statistics

Statistic	Description
Current Time	The time when this page was generated
Oracle Web Cache Start Timestamp	The time when Oracle Web Cache was started
Time Since Start	The length of time that Oracle Web Cache has been operating since it was started. Time is denoted in <i>days/hours/minutes/seconds</i> .
Total Number of Requests Served by Oracle Web Cache	Accumulated number of requests Oracle Web Cache has served since it was started See Also: "Gathering Oracle Web Cache Performance Statistics" on page 9-5 to view detailed statistics for Oracle Web Cache

Statistic	Description
Requests Served by Application Web Server Table	This table provides information about the number of requests served by the application Web servers. It contains the following columns: Requests Served by Application Web Servers: Name of the application Web server Up/Down: Specifies whether the application Web server is up or down Since: How long the application Web server has been up or down Total Request Served: Number of Web browser requests resolved by this application Web server Average Latency: Average amount of time for the Web browser requests to be resolved See Also: " Gathering Application Web Server Performance Statistics " on page 9-6 to view detailed statistics for application Web servers
Serving Requests/Second Now Bar	The health bar provides a graphical view of the number of Web browser requests resolved for each second by the: <ul style="list-style-type: none">▪ Documents in the cache that have expired or that have been invalidated, but have not yet been refreshed from the application Web servers▪ Documents in the cache that are still valid

Gathering Oracle Web Cache Performance Statistics

To monitor Oracle Web Cache performance:

1. Start Oracle Web Cache Manager.

See Also: "Starting Oracle Web Cache Manager" on page 4-2

2. In the navigator pane, select **Administering Web Sites > Monitoring Oracle Web Cache > Statistics**.

The Oracle Web Cache Statistics page appears in the right pane.

[Table 9-2](#) describes the statistics for this page.

Table 9-2 Oracle Web Cache Statistics

Statistic	Description
Last Modified	The time when this page was generated
Oracle Web Cache Start Timestamp	The time when Oracle Web Cache was started or restarted
Time Since Start	The length of time that Oracle Web Cache has been operating since it was started or restarted. Time is denoted in <i>days/hours/minutes/seconds</i> .
Number of Documents in Cache	Number of documents stored in Oracle Web Cache, plus the number of documents in transit through the cache
Cache Size (in bytes)	Current size of the cache Note: You can adjust the maximum size of the cache in the Maximum Cache Size page (Administering Oracle Web Cache > Max Cache Size).
Total Number of Bytes Written	Total number of bytes written to the cache
Current Number of Open Connections	Current number of incoming open connections to the Oracle Web Cache server and outgoing open connections to the application Web servers Note: You can adjust the limit of connections in the Resource Limits page (Administering Oracle Web Cache > Resource Limits).

Statistic	Description
Total Requests Served Table	<p>This table provides information about the number of requests Oracle Web Cache has or is currently serving to Web browsers:</p> <p>Number/Second Now: Total number of requests for each second currently being served by Oracle Web Cache</p> <p>Maximum/Second Since Start: Maximum number of requests for each second that Oracle Web Cache has served since it was started or restarted</p> <p>Average/Second Since Start: Average number of requests for each second that Oracle Web Cache has served since it was started or restarted</p> <p>Total Since Start: Accumulated number of requests that Oracle Web Cache has served since it was started or restarted</p>
Percentage Requests Served Table	<p>This table provides information about the percentage of requests that Oracle Web Cache is currently serving (% Now) and has served since it was started or restarted (% Since Start). It contains the following columns:</p> <p>Fresh Hits: Percentage of Web browser requests resolved by documents in the cache This percentage should be high, except when documents are being invalidated.</p> <p>Stale Hits: Percentage of Web browser requests resolved by documents that have expired or have been invalidated, but have not yet been retrieved from the application Web servers As documents are invalidated or expired, the percentage of stale hits will increase. The percentage will decrease as Oracle Web Cache retrieves updated content from the application Web servers. If the percentage does not decrease, it could indicate a bottleneck on the application Web servers.</p> <p>Refreshes: Percentage of documents Oracle Web Cache has refreshed from the application Web servers</p> <p>Cacheable Misses: Noncacheable Misses: Percentage of Web browser requests for noncacheable documents that were not served by Oracle Web Cache</p> <p>Noncacheable Misses: Percentage of Web browser requests for noncacheable documents that were not served by Oracle Web Cache</p>

Gathering Application Web Server Performance Statistics

To monitor application Web server performance:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager" on page 4-2](#)

2. In the navigator pane, select **Administering Web Sites > Monitoring Application Web Servers > Statistics**.

The Application Web Server Statistics page appears in the right pane.

[Table 9–3](#) describes the statistics for this page.

Table 9–3 Application Web Server Statistics

Statistic	Description
Application Web Server Statistics Table	<p>This table provides information about the application Web servers. It contains the following columns:</p> <ul style="list-style-type: none"> Application Web Server: Name of the application Web server Up/Down Time <ul style="list-style-type: none"> ■ Up/Down: Status of application Web server ■ Since: Time when the application Web server was started or stopped Completed Requests <ul style="list-style-type: none"> ■ Number/Sec: Number of requests that the application Web server is processing for each second ■ Max/Sec: Maximum number of requests that the application Web server can process for each second ■ Avg/Sec: Average number of requests that the application Web server has processed for each second ■ Total: Accumulated number of requests that the application Web server has processed Latency <ul style="list-style-type: none"> ■ Average this Interval: Average latency for 10 second intervals to process requests for Oracle Web Cache ■ Average Since Start: Average number of seconds to process requests for Oracle Web Cache since the application Web server started. Load <ul style="list-style-type: none"> ■ Now: Current number of connections from Oracle Web Cache that the application Web server has open ■ Max: Maximum number of connections that the application Web server has had open at one time <p>Note: If the number of Now connections is close to the Max connections, consider increasing the capacity of the application Web server. You can increase capacity in the Application Web Servers page (Administering Web Sites > Application Web Servers).</p> Active Sessions <ul style="list-style-type: none"> ■ Now: Current number of active connections from Oracle Web Cache to the application Web servers ■ Max: Maximum number of active connections that the application Web server has had open at one time

Statistic	Description
Apology Pages Served	# this second: Current number apology pages that Oracle Web Cache is serving to Web browsers, due to a network or busy Web site error Total: Total number of apology pages that Oracle Web Cache is serving to Web browsers, due to a network or busy Web site error
Application Web Server Backlog	Now: Current number of requests that the application Web server is processing for Oracle Web Cache Max: Maximum number of requests that the application Web server has processed for Oracle Web Cache

A

Oracle Web Cache Directory Structure

This appendix describes the installed Oracle Web Cache directory structure.

When you install Oracle Web Cache, all subdirectories are under a top-level `$ORACLE_HOME/webcache` directory on UNIX and an `ORACLE_HOME\webcache` directory on Windows NT. [Table A-1](#) describes the directory structure components.

Table A-1 Oracle Web Cache Directory Structure

Directory/File	Contents
<code>\$ORACLE_HOME/webcache/bin</code> on UNIX <code>ORACLE_HOME\bin</code> on Windows NT	Contains the Oracle Web Cache binaries, including the <code>webcached</code> main executable and the <code>webcachectl</code> command-line tool
<code>\$ORACLE_HOME/webcache/docs</code> on UNIX <code>ORACLE_HOME\webcache\docs</code> on Windows NT	Contains documentation and online help for the Oracle Web Cache Manager
<code>\$ORACLE_HOME/webcache/examples</code> on UNIX <code>ORACLE_HOME\webcache\examples</code> on Windows NT	Contains unsupported Oracle Web Cache scripts for invalidation and personalized attributes and system administration scripts for debugging. Read <code>README.examples.txt</code> in the directory for further information about the scripts.
<code>\$ORACLE_HOME/webcache/validation</code> on UNIX <code>ORACLE_HOME\webcache\validation</code> on Windows NT	Contains the Document Type Declaration (DTD) for invalidation requests
<code>\$ORACLE_HOME/webcache/logs</code> on UNIX <code>ORACLE_HOME\webcache\logs</code> on Windows NT	Contains event and access logs
<code>webcache.xml</code>	Configuration file that contains the configuration parameters set by the Oracle Web Cache Manager

B

Oracle Web Cache Default Settings

Oracle Web Cache is installed with several default settings that you can either use or modify. [Table B-1](#) describes the default configuration settings and where in the Oracle Web Cache Manager interface you can change the values.

Table B-1 Oracle Web Cache Default Settings

Configuration Settings	Default Value	Change Value
Security		
Password for the administrator user.	administrator	Administering Oracle Web Cache > Security
Password for the invalidator user.	invalidator	Administering Oracle Web Cache > Security
Process identify for Oracle Web Cache	User and group ID of user that installed Oracle Web Cache	Administering Oracle Web Cache > Process Identity
Ports		
Oracle Web Cache	1100	Administering Web Sites > Oracle Web Cache Listen Ports
Administration	4000	Administering Oracle Web Cache > Administration Port
Invalidation	4001	Administering Oracle Web Cache > Invalidation/Statistics Port
Statistics	4002	Administering Oracle Web Cache > Invalidation/Statistics Port

Configuration Settings	Default Value	Change Value
Application Web Server Failover		
Failover threshold	5	Administering Web Sites > Application Web servers
Polling interval for a failed application Web server	10 seconds	Administering Web Sites > Application Web servers
Logging		
Event logs	event_log in \$ORACLE_HOME/webcache/logs on UNIX and <i>ORACLE_HOME\webcache\logs</i> on Windows NT	This file name and default directory cannot be modified.
Access logs	access_log in \$ORACLE_HOME/webcache/logs on UNIX and <i>ORACLE_HOME\webcache\logs</i> on Windows NT	Administering Web Sites > Access Logging to modify the default directory location Note: The file name cannot be modified.
Resource Limits		
Maximum cache size	500 MB	Administering Oracle Web Cache > Resource Limits
Maximum incoming connections	1,000	Administering Oracle Web Cache > Resource Limits

C

Invalidation Document Type Declaration

This appendix describes the Document Type Declaration (DTD), or grammar, of invalidation requests and responses. It contains these topics:

- [Invalidation Request DTD](#)
- [Invalidation Response DTD](#)

Invalidation Request DTD

[Figure C-1](#) shows the DTD for the invalidation messages.

Figure C-1 Invalidation Request DTD

```
<!ELEMENT INVALIDATION (URL+)>

<!ELEMENT URL (VALIDITY, COOKIE*, HEADER*)>

<!ATTLIST URL
    EXP      CDATA #REQUIRED
    PREFIX   (YES|NO) "NO"
  >

<!ELEMENT VALIDITY EMPTY>
<!ATTLIST VALIDITY
    LEVEL   (0|1|2|3|4|5|6|7|8|9) "0"
    REFRESHTIME CDATA #IMPLIED
  >

<!ELEMENT COOKIE EMPTY>
<!ATTLIST COOKIE
    NAME    CDATA #REQUIRED
    VALUE   CDATA #IMPLIED
    NONEXIST (YES|NO) "NO"
  >

<!ELEMENT HEADER EMPTY>
<!ATTLIST HEADER
    NAME    CDATA #REQUIRED
    VALUE   CDATA #IMPLIED
  >
```

Table C–1 Invalidation Request DTD Elements and Attributes

Invalidation Element	Invalidation Attribute	Description
URL	EXP	URL of the document(s) that you requested to be invalidated. More than one URL element can be specified in the message.
	PREFIX	A value of YES invalidates documents that match a prefix-based regular expression. YES limits the URL expression (EXP) to a limited form of regular expression , whereby the URL path (from first "/" to the last "/") matches the prefix of the URL. For example, if you have the URL expression set to /cec/cstage\?ecaction=viewitem, then /cec/ is the common prefix of the URLs to invalidate. Therefore, URLs such as /cec/cstage?ecaction=viewitem&zip=94405 and /cec/cstage?ecaction=viewitem&zip=94305 match, but /usa/cec/cstage?ecaction=viewitem&zip=94209 does not match. The path section cannot contain any reserved regular expression characters. Reserved regular expression characters include periods (.), question marks (?), asterisks (*), brackets ([]), curly braces ({ }), carats (^), dollar signs (\$), or backslashes (\). However, the part after the path, which is usually the file name, can include regular expression characters. A value of NO (default) invalidates only documents contained within the URL. NO also means that the URL expression (EXP) is interpreted literally.
VALIDITY	LEVEL	Validity level of the documents. Validity determines how long Oracle Web Cache serves documents stale from the cache before removing them. The validity level ranges from 0 (for the least valid) to 9 (for the most valid). The higher the validity level, the longer Oracle Web Cache serves documents stale from the cache before removing them. Oracle Web Cache serves documents with a low validity level for a short amount of time before removing them. After documents are removed, Oracle Web Cache retrieves new versions of the documents from the application Web servers. A validity level of 0 means Oracle Web Cache will remove documents immediately. The default validity level is 0.
	REFRESHTIME	Maximum time that documents can reside in the cache

Invalidation Element	Invalidation Attribute	Description
COOKIE	NAME	Cookie name used by the documents contained within the URL. The name must match the cookie name of a multiple-version cacheability rule associated with this URL.
	VALUE	Value of the cookie
	NONEEXIST	A value of YES invalidates documents without this cookie; a value of NO invalidates documents with this cookie. NO is the default.
HEADER	NAME	HTTP request header used by the documents contained within the URL
	VALUE	Value of the request header. The name must match the header of a multiple-version cacheability rule associated with this URL.

Invalidation Response DTD

[Figure C-2](#) shows the DTD for the invalidation responses.

Figure C-2 Invalidation Response DTD

```
<!ELEMENT INVALIDATIONRESULT (URL+)>
<!ELEMENT URL    EMPTY>
<!ATTLIST URL
          EXPR      CDATA  #REQUIRED
          ID        CDATA  #REQUIRED
          STATUS    CDATA  #REQUIRED
          NUMINV   CDATA  #REQUIRED
>
```

Table C-2 Invalidation Response DTD Elements and Attributes

Invalidation Element	Invalidation Attribute	Description
URL	EXPR	URLs of the documents that you requested to be invalidated
	ID	Sequence number of all the URLs sent in the invalidation response. If there are multiple URLs specified in the invalidation message, then the sequence number starts at 1 for the first URL and sequentially numbers for each additional URL.
	STATUS	Status of the invalidation. Status can be one of the following:
		<ul style="list-style-type: none"> ▪ SUCCESS for successful invalidations ▪ URI NOT CACHEABLE for documents not in the cache ▪ URI NOT FOUND for documents not found
	NUMINV	Number of documents invalidated

D

Event Log Messages

This appendix describes the common information, warning, and error event log messages. It contains these topics:

- [Information Events](#)
- [Warning Events](#)
- [Error Events](#)

Information Events

Table D-1 lists the common event log informational messages.

Table D-1 Information Events

Message	Description
Startup Initialization Events	
Listening on ADMINISTRATOR <i>port</i> port address <i>ip_address</i>	The listening port number and IP address for administration requests.
Listening on INVALIDATION <i>port</i> port address <i>ip_address</i>	The listening port number and IP address for invalidation requests.
Listening on NORM <i>port</i> port address <i>ip_address</i>	The listening port number and IP address for Web browser requests to Oracle Web Cache.
Listening on STATISTICS <i>port</i> port address <i>ip_address</i>	The listening port number and IP address for statistics monitoring requests.
The cache server is started by the admin server at startup	The Oracle Web Cache <code>admin</code> process started the cache process.
The admin server started successfully	The Oracle Web Cache <code>admin</code> process successfully started.
The cache server started successfully	The Oracle Web Cache <code>cache</code> process successfully started.
Shutdown Events	
SIGTERM caught - program will shut down once all connections are complete.	A UNIX event that specifies that Oracle Web Cache will shut down once all connections are complete.
The server is exiting	Oracle Web Cache is shutting down.
Operational Events	
There was a network failure before the transaction was completed.	Oracle Web Cache terminated the connection to the Web browser.

Message	Description
Invalidation Events	
<Invalidation>Exact URI <i>URL</i> has been invalidated successfully.	The URL has been successfully invalidated.
<Invalidation>URI <i>URI</i> is not cacheable.	The URL is not cacheable document and cannot be invalidated.
<Invalidation> <i>number</i> URLs with prefix <i>prefix</i> have been successfully invalidated.	The number of URLs by a particular prefix that have been successfully invalidated.
<Invalidation>Requested URI <i>URI</i> is not found in the cache. <i>URI</i> is not invalidated.	The URL is not in the cache and cannot be invalidated.

Warning Events

[Table D-2](#) lists the common event log warning messages.

Table D-2 Warning Events

Message	Description
Startup Initialization Events	
The admin server couldn't start the cache server, running in admin-only mode.	The Oracle Web Cache admin process is unable to start the cache process. This may due to a listening port conflict.
Origin Server module got an ABORT; Errno: <i>error</i> URI: <i>URI</i>	Oracle Web Cache detects a problem with the application Web server, such as a shut down or failure.
Oracle Web Cache Memory-Related Events	
No space left for adding the Content-Length header for KeepAlive headers URI: <i>URI</i>	Oracle Web Cache does not have enough memory to allocate memory for Keep-Alive headers.
	For each response from the application Web server that does not contain a Content-Length field in the header, Oracle Web Cache allocates extra memory for Keep-Alive headers.
Response cookie header too large	The response cookie from the application Web server is too large for Oracle Web Cache.
Application Web Server Events	
<Admin Server>Concurrent administration exceeded limit	The number of concurrent connections (capacity) to the application Web servers has been exceeded
Connect Failed: Origin Web Server not accepting Connection	The application Web server is not accepting connections from Oracle Web Cache. This event could indicate that the application Web server is down.
Last-Modified time <i>time</i> is AFTER current time <i>time</i> , using current time instead	The Last-Modified field in the response header is after the current time. Oracle Web Cache will use the current time instead.
Expiration time beyond year 2038, setting to MAX_LONG <i>time</i> time overflow, setting to MAX_LONG	The time set for the document goes beyond the year 2038, which is the maximum time limit on Sun Solaris.

Error Events

Table D–3 lists the common event log error messages.

Table D–3 Error Events

Message	Description
Startup Initialization Events	
Failed to start the server.	Oracle Web Cache was unable to open listening ports.
Oracle Web Cache Cache failed to initialize.	Oracle Web Cache was unable to initialize cache module.
The server could not initialize	Oracle Web Cache was unable to start.
The server could not start service thread	Oracle Web Cache encountered a thread initialization creation error.
An error occurred scanning the directory <i>directory</i>	Oracle Web Cache encountered a problem while trying to load the Oracle Web Cache Manager help files and/or icons.
Read/Write Events	
Could not open config file (<i>config_file</i>)	Oracle Web Cache is unable to write to its configuration file due to a permissions problem.
Could not open log file (<i>access_log</i>)	Oracle Web Cache is unable to write to its access log file due to a permissions problem.
UNIX Process Identity Events	
Failed to find User (<i>user</i>) in /etc/password.	The current user cannot perform this operation. Only the root user or the user specified in the Process Identity page of the Oracle Web Cache Manager can perform this operation. (Administering Oracle Web Cache > Process Identity).
Invalid User ID (<i>user</i>).	
Failed to find Group (<i>group</i>) in /etc/group.	The group ID that the current user is a member of is not valid for this operation.
Invalid Group ID (<i>group</i>).	
Permission denied when setting User ID (<i>user</i>).	The current user that is being set is not the owner of the Oracle Web Cache executables and is not root user. Change the owner to the root user.
Permission denied when setting Group ID (<i>group</i>).	The current group that is being set is not the owner of the Oracle Web Cache executables.

Message	Description
Oracle Web Cache Memory-Related Events	
Cache failed to allocate memory for the hash table	Oracle Web Cache is unable to allocate memory for cache initialization.
Document compression error: <i>error</i>	Oracle Web Cache is unable to compress the document in its cache.
Cache Index memory allocation error	Oracle Web Cache is unable to allocate memory for cache initialization.
Out of memory	Oracle Web Cache is out of cache memory. The cache memory can be adjusted in the Resource Limits page of Oracle Web Cache Manager. (Administering Oracle Web Cache > Resource Limits).
Insert failed (memory low during insertion) for slave document <i>document</i> .	Oracle Web Cache is unable to insert a document into its cache.
The system has run critically low on memory	Oracle Web Cache is running critically low on cache memory. The cache memory can be adjusted in the Resource Limits page of Oracle Web Cache Manager. (Administering Oracle Web Cache > Resource Limits).
Operational Events	
Invalid XLF Field Name: <i>xlf_field</i>	The XLF field specified for the access log file is invalid. The XLF fields are specified in the Access Logs page of Oracle Web Cache Manager. (Administering Web Sites > Access Logs).
Too many session definitions	The number of allowed session definitions used for cacheability rules for pages with personalized attributes and/or session tracking has exceed the 20 name limit. Reduce the name of session names in Session/Personalized Attribute Definitions page of Oracle Web Cache Manager (Administering Web Sites > Session Management > Session/Personalized Attribute Definitions).

Message	Description
Application Web Server Events	
Unable to resolve the IP address of <i>ip_address</i> . Check your DNS setup.	Oracle Web Cache is unable to resolve the IP address of the application Web server. You can alter application Web server configuration in the Application Web Servers page of Oracle Web Cache Manager. (Administering Oracle Web Cache > Application Web Servers).
This product only supports IPv4 for origin server <i>ip_address</i> . Check your DNS setup.	Oracle Web Cache supports IP version 4. The IP address of the application Web server cannot be resolved because it uses another version of the IP.
Invalidation Events	
Invalidation Error: Default URL size too small for cache key.	The URL specified in the invalidation message is too long. Oracle Web Cache has a 3 KB limit on URLs that may or may not include cookies or HTTP request headers.
<Invalidation>Check ClientIP failed. Access denied.	The computer from which the invalidation message came from is not a trusted host. You configure trusted hosts in the Security page of Oracle Web Cache Manager. (Administering Oracle Web Cache > Security).
<Invalidation>Username/password check failed. Access denied.	The invalidation user name and password is not valid. The invalidation user is invalidator. By default, the password is invalidator. You can change the password in the Security page of the Oracle Web Cache Manager. (Administering Oracle Web Cache > Security).
<Invalidation>Empty entity.	The invalidation message is empty.
<Invalidation>Not an invalidation request.	The message is not an invalidation message.
<Invalidation>XML parsing error.	The invalidation message uses invalid XML syntax.
<Invalidation>Invalid validity level (valid range is 0-9). Level= <i>level</i> .	The validity level specified in the invalidation message is not valid.
<Invalidation>Cannot compose key pattern for the requested URI <i>URI</i> .	The URL specified in the invalidation message does not have a corresponding cacheability rule.
<Invalidation>Unrecognized cookies in the invalidation message.	The cookie(s) specified in the invalidation message are not valid.
<Invalidation>URL Node reading error.	The URL specified in the invalidation message is invalid or there is a memory allocation problem.

E

Troubleshooting Oracle Web Cache Configuration

This appendix describes common configuration problems and debugging techniques for resolving them. It contains these topics:

- [Startup Failures](#)
- [Application Web Server Capacity](#)
- [Wrong or Older Cached Content](#)

Startup Failures

If Oracle Web Cache does not start, it can be because of the following problems:

- [Port Conflicts](#)
- [Privileged Ports](#)
- [Cache Memory](#)

Port Conflicts

During configuration, you configure a listening port from which Oracle Web Cache receives browser requests. By default, this port is 1100. You also configure listening ports for administration, invalidation, and statistics monitoring requests. By default, these ports are 4000, 4001, and 4002, respectively. In addition to configuring listening ports for Oracle Web Cache, you also configure the advertised port number from which the application Web server(s) can receive Oracle Web Cache requests.

When you start Oracle Web Cache, a port conflict check is performed. If there is a port conflict, Oracle Web Cache will fail to start. Port conflicts are reported to the event log file, `event_log`. The `event_log` file is located in `$ORACLE_HOME/webcache/logs` on UNIX and in `ORACLE_HOME\webcache\logs` on Windows NT. [Figure E-1](#) shows an excerpt of `event_log` with port conflict event messages.

Figure E-1 Event Log with Port Conflict Messages

```
11/Jan/2001:11:04:42 -0800 -- Error: A failure occurred ( Address already in use ) when assigning a port ( domain: <NONE>, address: 0.0.0.0, port: 1100). Change PORT attribute of the LISTEN element in the configuration file to a suitable unused port.  
11/Jan/2001:11:04:42 -0800 -- Error: Failed to start the server.  
11/Jan/2001:11:04:42 -0800 -- Error: The server could not initialize  
11/Jan/2001:11:04:42 -0800 -- Information: The server is exiting  
11/Jan/2001:11:04:05 -0800 -- Warning: The admin server couldn't start the cache server, running in admin-only mode.
```

Note that the last message will only appear when the admin server process is started for the first time.

To resolve port conflicts:

1. Use Oracle Web Cache Manager to resolve the port conflicts.

Typically, the Oracle Web Cache and the application Web server ports are in conflict. Verify the port assigned to Oracle Web Cache at **Administering Web Sites > Oracle Web Cache Listen Port**, and verify the host names and ports assigned to the application Web servers at **Administering Web Sites > Application Web Servers**.

2. Restart Oracle Web Cache.

See Also: "Starting and Stopping Oracle Web Cache" on page 8-2

If the administration port is in conflict, the admin server process will not start, and the Oracle Web Cache Manager will not be accessible. The event log will contain event log messages that resemble the output in [Figure E-2](#).

Figure E-2 Event Log with Administration Port Conflict Messages

```
11/Jan/2001:10:56:11 -0800 -- Error: A failure occurred ( Address already in use ) when assigning a port ( domain: <NONE>, address: 0.0.0.0, port: 4000 ). Change PORT attribute of the LISTEN element in the configuration file to a suitable unused port.
11/Jan/2001:10:56:11 -0800 -- Error: Failed to start the server.
11/Jan/2001:10:56:11 -0800 -- Error: The server could not initialize
11/Jan/2001:10:56:11 -0800 -- Information: The server is exiting
```

To resolve this port conflict, modify the webcache.xml file, an internal file that contains the configuration settings, and change the administration port number. The webcache.xml file is located in \$ORACLE_HOME/webcache on UNIX and in ORACLE_HOME\webcache on Windows NT. [Figure E-3](#) shows an excerpt of the webcache.xml file with the line for the administration port shown in boldface.

Figure E-3 webcache.xml File

```
<?xml version="1.0"?>
<!DOCTYPE CALYPSO SYSTEM "internal:///webcache.dtd">
<CALYPSO>
  <MULTIPORT>
    <LISTEN IPADDR="ANY" PORT="1100" PORTTYPE="NORM"/>
    <LISTEN IPADDR="ANY" PORT="4000" PORTTYPE="ADMINISTRATION">
    <LISTEN IPADDR="ANY" PORT="4003" PORTTYPE="INVALIDATION"/>
    <LISTEN IPADDR="ANY" PORT="4002" PORTTYPE="STATISTICS"/>
  </MULTIPORT>
```

Cache Memory

Oracle Web Cache pre-allocates a large memory pool for data storage. When Oracle Web Cache is started, the admin and the cache server processes require 200 MB of memory to start. If there is not enough physical or virtual memory for these processes, the cache server process will fail to start. The event log will contain event log messages that resemble the output in [Figure E-4](#).

Figure E-4 Event Log with Cache Memory Messages

```
10/Oct/2000:10:58:02 -0600 -- Error: Oracle Web Cache Cache failed to initialize  
10/Oct/2000:10:58:02 -0600 -- Error: The server could not initialize  
10/Oct/2000:10:58:02 -0600 -- Information: The server is exiting  
10/Oct/2000:10:58:02 -0600 -- Warning: The admin server couldn't start the cache  
server, running in admin-only mode
```

To resolve this cache memory issue:

1. Allocate additional memory for the admin and the cache server processes to start.
2. Restart Oracle Web Cache.

See Also: ["Starting and Stopping Oracle Web Cache"](#) on page 8-2

Privileged Ports

Port numbers below 1024 are reserved for use by privileged processes on UNIX. If you want to configure Oracle Web Cache to listen on a port below 1024, such as on port 80, change the ownership of the webcached executable to `root` and run it as `root`. By default, the user that performed the installation is the owner of Oracle Web Cache executable.

To configure a privileged port:

1. Start Oracle Web Cache Manager.

See Also: ["Starting Oracle Web Cache Manager" on page 4-2](#)

2. Configure the privileged port:

- a. In the navigator pane, select **Administering Web Sites > Oracle Web Cache Listen Ports**.

The Oracle Web Cache Listen Ports page appears in the right pane.

- b. In the Oracle Web Cache Listen Ports page, choose **Add**.

The Edit/Create Web Cache Listen Ports page dialog box appears.

- c. In the **Oracle Web Cache IP Address** field, enter the IP address of the computer running Oracle Web Cache.

- d. In the **Oracle Web Cache Listening Port** field, enter the listening port from which Oracle Web Cache will receive Web browser requests for the Web site. Ensure this port number is not already in use.

- e. Choose **Submit**.

3. Change the process identify of the webcached executables.

If Oracle Web Cache was installed by the `root` user, change the user ID and group ID to the desired running process identify for Oracle Web Cache. The webcached executable will still be owned by `root`, but it will take on the new process identify once the privileged port is opened.

If Oracle Web Cache was installed by a non-root user, you may want to change the user ID and group ID to a different user and group ID, such as `nobody/nobody`.

To change the process identity:

- a. In the navigator pane, select **Administering Oracle Web Cache > Process Identity**.
The Process Identity page appears in the right pane.
 - b. In the Process Identity page, choose **Change IDs**.
The Change Process Identity dialog box appears.
 - c. Enter the new user in the **New User ID** field and the group ID of the user in the **New Group ID** field.
 - d. Choose **Submit**.
4. In the Oracle Web Cache Manager main window, choose **Apply Changes**.
 5. If not already owned by `root`, change the ownership of the `webcached` executable to `root`.
 - a. Exit from Oracle Web Cache Manager, stop Oracle Web Cache, and log out of the computer.

See Also: ["Starting and Stopping Oracle Web Cache" on page 8-2](#)

- b. Log back in as `root`.
- c. Change the ownership of the executable `webcached` to `root`.

```
chown root $ORACLE_HOME/webcache/bin/webcached
```

6. Change the group of the executable `webcached` to the group ID you configured in Step 3.

```
chgrp group_id $ORACLE_HOME/webcache/bin/webcached
```

7. Add set-user ID permission to the `webcached` executable.

```
chmod u+s $ORACLE_HOME/webcache/bin/webcached
```

8. Log out of the computer, and re-login as the user configured in the Process Identify page.

9. Start Oracle Web Cache.

When Oracle Web Cache starts, it will listen on the privileged port with the new process identity.

Application Web Server Capacity

If an application Web server has reached capacity the following error message appears when accessing pages of a Web site:

The application Web server is busy. Possible reach capacity.

This error indicates that the application Web server has reached capacity—that is, the number of concurrent connections has been exceeded. To resolve this problem, you can either:

- Increase capacity

In the **Administering Oracle Web Cache > Resource Limits** page of Oracle Web Cache Manager, check the value of the **Current Maximum incoming connections** field. This field provides the currently configured capacity. If the capacity can be adjusted, increase it.

See Also: ["Task 3: Set Resource Limits" on page 5-5](#)

- Evaluate the caching rules to determine if additional content can be cached

See Also: [Chapter 6, "Creating Rules for Cached Content"](#)

Wrong or Older Cached Content

If Oracle Web Cache is serving wrong or older content, perform these steps:

1. Check the correctness of rules.
2. Check the precedence, or order, of rules.

See Also: [Chapter 6, "Creating Rules for Cached Content"](#)

3. Validate caching rules by analyzing the content of the access log file, `access_log`, and the event log file, `event_log`. If verbose logging is turned on in the event log, error messages about the cacheability rules will be reported.

See Also: ["Configuring Event Logs" on page 8-21](#) for further information about turning on verbose logging

Configuration Changes Made in Oracle Web Cache Manager

Configuration changes made in Oracle Web Cache Manager require the following steps:

1. In the Oracle Web Cache Manager main window, choose **Apply Changes**.
2. Stop and then restart Oracle Web Cache.

See Also: ["Starting and Stopping Oracle Web Cache" on page 8-2](#)

If these steps are not followed, configuration changes will not take effect.

The currently displayed status message in Oracle Web Cache Manager informs you if one of these steps needs to be performed, or if the Oracle Web Cache is running with the current configuration.

See Also: ["Status Messages" on page 4-4](#) for further information about status message in Oracle Web Cache Manager

Glossary

access log

A log file that contains information about the HTTP requests sent to Oracle Web Cache for a Web site. The access log has a file name of `access_log` and is stored by default in `$ORACLE_HOME/webcache/logs` and `ORACLE_HOME\webcache\logs` on Windows NT.

application Web server

A server that manages data for a Web site, controls access to that data, and responds to requests from Web browsers. The application on the Web server interfaces with the database and performs the job requested by the Web server.

cache hit

A HTTP Web browser request that can be satisfied from the Oracle Web Cache cache without going to the application Web server.

cache miss

A HTTP Web browser request that cannot be satisfied from the Oracle Web Cache cache and most go to the application Web server.

category cookie

A **cookie** that enables the multiple version of the same page to served to different categories of users.

CLF

See [Common LogFile Format \(CLF\)](#).

Common LogFile Format (CLF)

An industry-standard format for Web transaction log files.

cookie

A packet of state information sent by an application Web server to a Web browser during a HTTP request. During subsequent HTTP requests, the cookie is passed back to the application Web server, enabling the application Web server to remember the state of the last transaction. Some uses of cookies include:

- Identifying a registered user
- Maintaining a shopping cart selected during a session
- Session tracking

DNS

See [Domain Name System \(DNS\)](#).

Domain Name System (DNS)

A system for naming computers and network services that is organized into a hierarchy of domains. DNS is used in TCP/IP networks to locate computers through user-friendly names. DNS resolves a friendly name into an [IP address](#), which is understood by computers.

Document Type Declaration (DTD)

Markup declarations that provide a grammar for a class of documents.

event log

A log file that contains Oracle Web Cache event and error information. The event log has a file name of `error_log` and is stored in `$ORACLE_HOME/webcache/logs` on UNIX and `ORACLE_HOME\webcache\logs` on Windows NT.

expiration

Time when documents are no longer valid in the cache and are refreshed.

Extended LogFile Format (XLF)

An improved format for HTTP server logins since it is extensible, permitting a wider range of data to be captured. XLF allows you to configure the logger to generate different statistics of HTTP requests such as the IP address of clients, methods of the HTTP requests and response headers such as user agent and accept.

Extensible Markup Language (XML)

A language that offers a flexible way to create common information formats. XML is used for invalidation messages and responses.

failover

When an application Web server fails, Oracle Web Cache automatically distributes the load over the remaining application Web servers and polls the failed application Web server for its current up/down status every 60 seconds until it is back online.

GET method

A HTTP method used for simple requests for Web pages. A GET method is made up of a URL. Requests for pages that use the GET method are typically cached.

GET method with query string

A HTTP method made up of a URL and a query string containing parameters and values. An example of a HTTP GET with query string follows.

`http://www.myserver.com/setup/config/navframe?frame=default`

This request executes a script named navframe in the /setup/config directory of the www.myserver.com server and passes the script a value of default for the frame variable.

Note: You should not cache pages with GET with query strings forms that make changes to the application Web server(s) or database. You should only cache pages that use GET with query strings if they are used in searches.

HTTP protocol

Hypertext Transport Protocol. A protocol that provides the language that enables browsers and application Web servers to communicate.

HTTP request header

A header that enables Web browsers to pass additional information about the request and about itself to the application Web server.

HTTP request method

A method included in the HTTP request that specifies the purpose of the client's request. HTTP supports many methods, but the ones that concern caching are GET, GET with query string, and POST methods.

HTTPS protocol

Secure Hypertext Transfer Protocol. A protocol that uses the Secure Sockets Layer (SSL) to encrypt and decrypt user page requests as well as the pages that are returned by the application Web server.

invalidation

Process that marks documents as invalid and then refreshes them with updated content from the application Web servers. Invalidation keeps the Oracle Web Cache cache consistent with the content on the application Web servers.

IP address

Used to identify a node on a network. Each computer on the network is assigned a unique IP address, which is made up of the network ID, and a unique host ID. This address is typically represented in dotted-decimal notation, with the decimal value of each octet separated by a period, for example 144.45.9.22.

latency

Networking roundtrip time.

load balancing

A feature in which HTTP requests are distributed among application Web servers so that no single server is overloaded.

Layer 4 (L4) switch

A networking switch that operates at Layer 4 of the **Open Systems Interconnection (OSI)** model—the Transport layer. L4 switches base their switching decisions on the TCP/IP protocol header and determine, based on the port number, where to pass traffic.

Layer 7 (L7) switch

A networking switch that operates at Layer 7 of the OSI model—the Application layer. L7 switches base their switching decisions on URL content.

Open Systems Interconnection (OSI)

A model of network architecture developed by ISO as a framework for international standards in heterogeneous computer network architecture.

The OSI architecture is split between seven layers, from lowest to highest:

1. Physical layer
2. Data link layer
3. Network layer
4. Transport layer
5. Session layer
6. Presentation layer
7. Application layer

Each layer uses the layer immediately below it and provides a service to the layer above.

OSI

See [**Open Systems Interconnection \(OSI\)**](#).

Oracle Web Cache Manager

A graphical user interface tool that combines configuration abilities with component control to provide an integrated environment for configuring and managing Oracle Web Cache.

performance assurance heuristics

Heuristics that enable Oracle Web Cache to assign a queue order to documents. These heuristics determine which documents can be served stale and which documents must be retrieved immediately. While documents with a higher priority are retrieved first, documents with a lower priority are retrieved at a later time.

The queue order of documents is based on the popularity of documents and the validity of documents assigned during invalidation. If the current load and capacity of the application Web server is not exceeded, the most popular and least valid documents are refreshed first.

personalized attributes

Pages that contain personalized attributes, such as personalized greetings in the form of "Welcome <your name>", icons, addresses, or shopping cart snippets, on an otherwise generic page. You can configure Oracle Web Cache to cache the instructions for substituting values for personalized attributes based on the information contained within a **cookie** or an embedded URL parameter.

popularity

The number of requests for a document since entering the cache and the number of recent requests for the document.

POST method

A HTTP method used for requests that modify the contents of the data store on the application Web server, such as posting a message to a mailing list, submitting forms for registration purposes, or adding entries to the database.

Note: You should not cache pages with POST forms that make changes to the application Web server(s) or database. You should only cache pages that use POST forms if they are used in searches.

regular expression

Oracle Web Cache supports the POSIX 1003 extended regular expressions for URLs, as supported by Netscape Proxy Server 2.5.

See Also:

http://www.cs.utah.edu/dept/old/texinfo/regex/reg_ex_toc.html for regular expression syntax

reverse proxy server

A proxy server that appears to be a normal server to browsers but internally retrieves its documents from other application Web servers as a proxy.

session binding

The process of binding a user session to a given application Web server in order to maintain state for a period of time.

session cookie

A **cookie** that enables a Web site to keep track of user sessions. Session cookies are used for **session binding** and **session tracking**.

session tracking

Session information passed back and forth between a Web browser and an application Web server. This is typically done with a unique sequential number and/or cookie.

session-encoded URLs

 HTML tags containing session information. Session-encoded URLs enable Web sites to keep track of user sessions. Oracle Web Cache can cache the instructions for replacing session information for one user with another based on the personal information contained within a cookie or as an embedded parameter in the URL.

Uniform Resource Identifier (URI)

The address syntax that is used to create **URLs**.

Uniform Resource Locator (URL)

A standard for specifying the location and route to a file on the Internet. URLs are used by browsers to navigate the World Wide Web and consist of a protocol, domain name, directory path, and the file name. For example, <http://otn.oracle.com/products/ias> specifies the location and path a browser will travel to find the Oracle Technology Network's Oracle9i Application Server site on the World Wide Web.

URI

See **Uniform Resource Identifier (URI)**.

URL

See **Uniform Resource Locator (URL)**.

validity

A level of validity assigned during **invalidation** or **expiration**.

The higher the validity level, the longer Oracle Web Cache serves documents stale from the cache before removing them. Oracle Web Cache serves documents with a low validity level for a short amount of time before removing them. After

documents are removed, Oracle Web Cache retrieves new versions of the documents from the application Web servers.

Critical documents should be assigned a low validity level, and non-critical documents should be assigned a high validity level.

XLF

See [ExtendedLogFile Format \(XLF\)](#).

XML

See [Extensible Markup Language \(XML\)](#).

Index

Symbols

\$ (dollar sign) symbol
 regular expression, 6-3, 8-7, 8-12, C-3
* (asterisk) symbol
 regular expression, 6-3, 8-7, 8-12, C-3
? (question mark) symbol
 regular expression, 6-3, 8-7, 8-12, C-3
[] (brackets) symbol
 regular expression, 8-7, 8-12, C-3
\ (backslash) symbol
 regular expression, 6-3, 8-7, 8-12, C-3
^ (carat) symbol
 regular expression, 6-3, 8-7, 8-12, C-3
{ } (braces) symbol
 regular expression, 8-7, 8-12, C-3
. (period) symbol
 regular expression, 6-3, 8-7, 8-12, C-3

Numerics

1024 port, E-5
1100 port, 5-9, B-1
4000 port, B-1
4001 port, 8-4, B-1
4002 port, 9-2, B-1

A

Accept request header, 2-8
Accept-Charset request header, 2-8
Accept-Encoding request header, 2-8
Accept-Language request header, 2-8

access logs
 configuring settings for, 8-27
 described, 8-22
 disabling, 8-28
 examples of, 8-24 to 8-26
 format of, 8-23
 XLF fields, 8-23
access_log file, B-2
access_logyyyymmdd, 8-27
Active Server Pages (ASP), 1-7, 2-5
admin server process
 described, 8-2
 restriction, E-4
administrator user, 5-2, B-1
application Web servers
 capacity, 5-7
 concurrent connections, 2-3
 configuring for Oracle Web Cache, 5-7
failover, 1-10
 connection request threshold, 5-8
 polling failed servers, 5-9
load balancing
 configuration, 7-2
 described, 1-9
load of, 2-3
performance assurance, 2-3
performance monitoring, 9-6
session binding
 configuring, 7-3
 described, 1-12
Application Web Servers page in Oracle Web Cache
 Manager, 5-7
authoritative DNS server, 3-16

B

backend failover, 1-10
bin directory, A-1
binding, 1-12

C

Cache Cleanup page in Oracle Cache Manager, 8-11

cache hit, 1-4
cache memory configuration, 5-5
cache memory, troubleshooting of, E-4
cache miss, 1-4
cache population, 2-2
cache server process
described, 8-2
memory restriction, E-4
cache size configuration, 5-5
cacheability rules, 6-5 to 6-9
default, 6-4
overview, 6-2
PDF documents, 6-4
capacity
described, 5-7
troubleshooting of, E-7

category cookies
described, 2-7
request and response value comparison, 2-7

Common Gateway Interface (CGI), 1-7, 2-5

compression
configuration, 6-8
described, 1-14

configuring
access logs, 8-27
application Web server settings, 5-7
cacheability rules, 6-5 to 6-9
cacheability rules for HTTP error codes, 6-8
cacheability rules for multiple versions of the same URL
cookie values, 6-6, 6-14
HTTP request headers, 6-15
cacheability rules for personalized attributes, 6-8, 6-16
cacheability rules for session tracking, 6-8, 6-24
cacheability rules for session-encoded URLs, 6-8, 6-16

compression, 6-8
event logs, 8-21
expiration rules, 6-5, 6-12
failover of application Web servers, 7-2
load balancing of application Web server, 7-2
quick reference to procedures, 4-7
resource limits, 5-5
security settings, 5-2
session binding to an application Web server, 7-3
COOKIE invalidation message element, 8-9, C-4
cookies
cacheability rules
documents with category cookies, 6-6, 6-14
documents with personalized attributes, 6-8, 6-16
documents with session cookies, 6-8, 6-24
category cookies for multiple versions of the same URL, 2-7
described, 1-12
personalized attributes, 2-9
session cookies
for session binding, 1-12

D

deploying Oracle Web Cache
for part of a Web site, 3-8
in a distributed network, 3-14
in a failover pair, 3-10
inside a firewall, 3-12
outside a firewall, 3-13
with HTTPS requests, 3-4
with multiple application Web servers, 3-6
with one application Web server, 3-2

directory structure
bin directory, A-1
docs directory, A-1
examples directory, A-1
invalidation directory, A-1
logs directory, A-1
DNS server, 1-4
docs directory, A-1

dynamically generated content caching, 1-7
Active Server Pages (ASP), 1-7, 2-5
Common Gateway Interface (CGI), 1-7, 2-5
described, 2-5
Java Server Pages (JSP), 1-7, 2-5
Java Servlets, 1-7, 2-5
multiple versions of the same URL, 2-6
personalized attributes, 2-9
personalized greetings, 2-9
PL/SQL Server Pages (PSP), 1-7, 2-5
session tracking, 2-11
session-encoded URLs, 2-12

E

error messages in event log, D-5
event logs
configuring settings for, 8-21
described, 8-19
error messages, D-5
examples of, 8-19 to 8-21
finding errors, 8-21
format of, 8-19
information messages, D-2
warning messages, D-4
event_log file, B-2
examples directory, A-1
EXP invalidation message attribute, 8-7, C-3
EXP invalidation response attribute, 8-10, C-5
expiration
concepts of, 2-2
performance assurance heuristics, 6-13
validity, 6-13
expiration rules, 6-5, 6-12
by cache entry, 6-12
by document creation, 6-12
by HTTP Expires header, 6-12
Expires header, 6-12

F

failover
configuring, 7-2
connection request threshold, 5-8
described, 1-10
polling failed application Web servers, 5-9
firewalls and Oracle Web Cache deployments, 3-13

G

GET methods, 6-5
GET with query string method, 6-5
grep command, 8-21
group ID for Oracle Web Cache
administration, 5-4

H

HEADER invalidation message element, 8-9, C-4
HTTP error code cacheability rules, 6-8
HTTP request headers, 2-7, 6-7
Accept, 2-8
Accept-Charset, 2-8
Accept-Encoding, 2-8
Accept-Language, 2-8
cacheability rules for, 6-7, 6-15
supported by Oracle Web Cache, 2-8
User-Agent, 2-8
HTTPS requests, 3-4

I

ID invalidation response attribute, 8-10, C-5
information messages in event log, D-2
Invalidate.java file, 8-15
invalidating documents
database triggers, 8-15
HTTP POST messages, 8-6
Oracle Web Cache Manager, 8-11
invalidation
concepts of, 2-2
described, 1-8
performance assurance heuristics, 8-14
port number, 8-4, 9-2
user, 5-2
validity, 8-9, 8-14
invalidation directory, A-1
invalidation messages
COOKIE element, 8-9, C-4
EXP attribute, 8-7, C-3
HEADER element, 8-9, C-4
LEVEL attribute, 8-9, C-3
NAME attribute, 8-9, C-4
NONEXIST attribute, 8-9, C-4
PREFIX attribute, 8-7, C-3
REFRESHTIME attribute, 8-9, C-3

regular expression
. (period) symbol, 8-7, 8-12, C-3
\$ (dollar sign) symbol, 8-7, 8-12, C-3
* (asterisk) symbol, 8-7, 8-12, C-3
? (question mark) symbol, 8-7, 8-12, C-3
[] (brackets) symbol, 8-7, 8-12, C-3
\ (backslash) symbol, 8-7, 8-12, C-3
^ (carat), 8-7, 8-12, C-3
{ } (braces) symbol, 8-7, 8-12, C-3
syntax of, 8-7
URL element, 8-7, C-3
VALIDITY element, 8-9, C-3
VALUE attribute, 8-9, C-4
invalidation responses, 8-10, C-5
 EXP attribute, 8-10, C-5
 ID attribute, 8-10, C-5
 NUMINV attribute, 8-10, C-5
 STATUS attribute, 8-10, C-5
 syntax of, 8-7
 URL element, 8-10, C-5
invalidator user, 5-2, B-1

J

Java Server Pages (JSP), 1-7, 2-5
Java Servlets, 1-7, 2-5

L

L4 (Layer 4) switches
 HTTP requests, 3-4
L7 (Layer 7) switch, 3-8
LEVEL invalidation message attribute, 8-9, C-3
load balancing

M

multiple versions of the same URL, 2-6
 cookie values, 2-7
 HTTP request headers, 2-7
multiple versions of the same URL cacheability rules
 cookie values, 6-7, 6-14

N

NAME invalidation message attribute, 8-9, C-4
netstat command, 5-5
NONEEXIST invalidation message attribute, 8-9, C-4
NUMINV invalidation response attribute, 8-10, C-5

O

Oracle Web Cache
 admin server process, 8-2
 administration, 1-14
 benefits
 cost savings, 1-6

Oracle Web Cache Manager

administering Oracle Web Cache

- event logs, 8-21
- invalidating documents, 8-11
- resource limits, 5-5
- security settings, 5-2

administering Web sites

- access logs, 8-27
- application Web server settings, 5-7
- cacheability rules, 6-5 to 6-9
- cacheability rules for HTTP error codes, 6-8
- cacheability rules for multiple versions of the same URL, 6-6, 6-7, 6-14, 6-15
- cacheability rules for personalized attributes, 6-8, 6-16
- cacheability rules for session tracking, 6-8, 6-24
- cacheability rules for session-encoded URLs, 6-8, 6-16
- compression, 6-8
- expiration rules, 6-5, 6-12
- load balancing, 7-2
- session binding, 7-3

applying changes, E-8

described, 4-2

introduced, 1-14

monitoring application Web servers

- performance statistics, 9-6

monitoring Oracle Web Cache

- cache health, 9-3
- performance statistics, 9-5

navigator pane, 4-5

restarting Oracle Web Cache, 5-10

right pane, 4-6

starting, 4-2

starting Oracle Web Cache, 8-2

status messages, 4-4

Oracle9i Application Server with Oracle Web Cache, 1-2

OracleHOME_NAMEWebCache service, 8-3

OracleHOME_NAMEWebCacheAdmin service, 8-3

P

PDF documents, 6-4

performance assurance

- described, 2-2

- introduced, 1-8

performance assurance heuristics, 1-8, 2-3

- application Web server limit, 2-3

- application Web server load, 2-3

- concurrent connections, 2-3

- expiration, 6-13

- invalidation, 8-14

- popularity, 2-3

- validity, 2-3

performance monitoring

- application Web servers, 9-6

- Oracle Web Cache, 9-5

personalized attribute cacheability rules, 6-8, 6-16

personalized attributes

- WEBCACHEEND HTML tag, 2-9, 6-17

- WEBCACHETAG HTML tag, 2-9, 6-17

personalized greetings

- see personalized attributes

PL/SQL Server Pages (PSP), 1-7, 2-5

- popularity, 2-3

populating the cache, 2-2

port conflicts, E-2

ports

- 1024, E-5

- 1100, 5-9, B-1

- 4000, B-1

- 4001, 8-4, B-1

- 4002, 9-2, B-1

POSIX 1003 extended regular expressions, 6-3

POST methods, 6-5

PREFIX invalidation message attribute, 8-7, C-3

privileged ports, E-5

R

README.examples.txt file, 8-15, A-1

REFRESHTIME invalidation message

- attribute, 8-9, C-3

regular expression, 6-3
 . (period) symbol, 6-3, 8-7, 8-12, C-3
 \$ (dollar sign) symbol, 6-3, 8-7, 8-12, C-3
 * (asterisk) symbol, 6-3, 8-7, 8-12, C-3
 ? (question mark) symbol, 6-3, 8-7, 8-12, C-3
 [] (brackets) symbol, 8-7, 8-12, C-3
 \ (backslash) symbol, 6-3, 8-7, 8-12, C-3
 ^ (carat) symbol, 6-3, 8-7, 8-12, C-3
 { } (braces) symbol, 8-7, 8-12, C-3
Resource Limits page in Oracle Web Cache
 Manager, 5-6
restarting Oracle Web Cache, 5-10
reverse proxy server, 1-2
rules for creating cacheability rules, 6-2

S

security, 1-14
Security page in Oracle Web Cache Manager, 5-2
security settings, 5-2
session binding
 configuring, 7-3
 described, 1-12
session cookies
 described, 2-11
 request and response value comparison, 2-11,
 6-24
session tracking
 cacheability rules for, 6-8, 6-24
 described, 2-11
 serving the first request from the cache, 6-27
session-encoded URLs
 cacheability rules for, 6-8, 6-16
 described, 2-12
starting
 Oracle Web Cache, 5-2, 8-2
 Oracle Web Cache Manager, 4-2
startup failures, E-2
static content caching, 1-7
STATUS invalidation response attribute, 8-10, C-5
status messages in Oracle Web Cache Manager, 4-4
stopping Oracle Web Cache, 8-2
surge protection, 1-9

T

troubleshooting
 application Web server capacity, E-7
 cache memory, E-4
 caching PDF documents, 6-4
 GMT to local timestamp, 8-22, 8-27
 Oracle Web Cache Manager, E-8
 port conflicts, E-2
 privileged ports, E-5
 startup failures, E-2
 wrong or older cached content, E-7
trusted subnet for Oracle Web Cache
 administration, 5-3

U

URL invalidation message element, 8-7, C-3
URL invalidation response element, 8-10, C-5
user ID for Oracle Web Cache administration, 5-4
User-Agent request header, 2-8
utl_proc.sql script, 8-15
UTL_TCP Oracle supplied package, 8-15

V

validity, 2-3
validity for expiration, 6-13
validity for invalidation, 8-9, 8-14
VALIDITY invalidation message element, 8-9, C-3
VALUE invalidation message attribute, 8-9, C-4

W

warning messages in event log, D-4
Web caching
 benefits
 cost savings, 1-6
 high availability, 1-6
 network traffic reduction, 1-6
 performance, 1-5
 scalability, 1-5
 described, 1-3
webcachectl start command, 4-6, 5-2, 5-10
webcachectl status command, 4-6, 8-2
webcachectl stop command, 4-6, 5-10

webcachectl utility
 obtaining the status of Oracle Web Cache, 8-2
 restarting Oracle Web Cache, 5-10
 starting Oracle Web Cache, 8-2
 stopping Oracle Web Cache, 8-2
webcachectl utility syntax, 4-6
webcached executable, A-1
WEBCACHEEND HTML tag for personalized
 attributes, 2-9, 6-17
WEBCACHETAG HTML tag for personalized
 attributes, 2-9, 6-17
webcache.xml, A-1
webcache.xml file, E-3

X

XLF fields for access logs, 8-23

