

Oracle9iAS Web Cache

Administration and Deployment Guide

Release 2.0.0

October 10, 2001

Part No. A90372-04

ORACLE®

Part No. A90372-04

Copyright © 2000, 2001, Oracle Corporation. All rights reserved.

Primary Author: Deborah Steiner

Contributors: Jose Alvarez, Steve Andrew, Jesse Anton, Christine Chan, Ajay Dsai, Jay Feenan, Patrick Fry, Pramodini Gattu, Helen Grembowicz, Hideaki Hayashi, Martin Littlecott, Larry Jacobs, Lars Klevan, Xiang Liu, Rajiv Mishra, Jordan Parker, Marcin Porwit, Charles Qi, Parthiban Thilagar, William Wright, Jean Zeng, Tie Zhong, and Yuhui Zhu

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 Oracle9i and Oracle8i are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

Regular Expression Rights and License Notice The Windows release of Oracle9iAS Web Cache includes source code from the regular expression (regex) directory of release 4.01 of the PHP source distribution from the PHP Group. The regex source code is a copyright of Henry Spencer and licensed from the PHP Group. The following applies only to the regex code which is compiled and linked in the webcached.exe binary.

Henry Spencer Copyright Notice

Copyright © 1992, 1993, 1994, Henry Spencer. All rights reserved.

This software is not subject to any license of the American Telephone and Telegraph Company or of the Regents of the University of California.

Permission is granted to anyone to use this software for any purpose on any computer system, and to alter it and redistribute it, subject to the following restrictions:

1. The author is not responsible for the consequences of use of this software, no matter how awful, even if they arise from flaws in it.
2. The origin of this software must not be misrepresented, either by explicit claim or by omission. Since few users ever read sources, credits must appear in the documentation.
3. Altered versions must be plainly marked as such, and must not be misrepresented as being the original software. Since few users ever read sources, credits must appear in the documentation.
4. This notice may not be removed or altered.

PHP License Agreement for regex Source Code

Copyright © 1999, 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	xvii
Preface.....	xix
Audience	xx
Organization.....	xx
Related Documentation	xxii
Conventions.....	xxiv
Documentation Accessibility	xxviii
What's New in Oracle Web Cache?.....	xxix
Part I Getting Started with Oracle Web Cache	
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	1-4
Benefits of Web Caching.....	1-5
Features of Oracle Web Cache	1-7
Static and Dynamically Generated Content Caching.....	1-8
Cache Invalidation.....	1-9
Performance Assurance	1-9
Surge Protection of Application Web Servers	1-10

Load Balancing of Application Web Servers	1-10
Backend Failover	1-12
Application Web Server Binding.....	1-14
Security Features.....	1-16
Restricted Administration	1-16
Secure Sockets Layer (SSL) Support	1-16
Administration.....	1-19
Compression.....	1-19

2 Oracle Web Cache Concepts

Populating Oracle Web Cache	2-2
Request and Response Header Fields.....	2-3
Cache Freshness and Performance Assurance.....	2-4
Caching Dynamically Generated Content	2-7
Multiple Versions of the Same Document	2-8
Personalized Attributes	2-12
Session Information.....	2-14
Session Tracking	2-14
Session-Encoded URLs	2-15
Content Assembly and Partial Page Caching.....	2-17
Page Assembly Components	2-18
ESI Features	2-22
ESI for Java (JESI).....	2-22

3 Deploying Oracle Web Cache

Caching Content for One Application Web Server.....	3-2
Caching Content for HTTPS Requests	3-4
Load Balancing Requests Among Application Web Servers	3-9
Accelerating Portions of a Web Site	3-10
Using Oracle Web Cache Servers in a Failover Pair.....	3-12
Working with Firewalls	3-14
Deploying Oracle Web Cache Servers in a Distributed Network.....	3-17

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	4-7
Configuration and Administration Tasks at a Glance	4-7

Part II Configuration and Administration of Oracle Web Cache

5 Initial Setup and Configuration

Task 1: Start Oracle Web Cache	5-2
Task 2: Modify Security Settings	5-2
Task 3: Specify Web Site Settings	5-5
Task 4: Set Resource Limits	5-8
Cache Memory	5-8
Connection Limit	5-12
Connections on UNIX.....	5-12
Connections on Windows	5-14
Task 5: (Optional) Modify Ports for Administration, Invalidation, and Statistics Monitoring Requests	5-15
Changing the Default Administration Listening Endpoint	5-15
Changing the Default Invalidation and Statistics Monitoring Listening Endpoints.....	5-16
Task 6: (Optional) Configure the Oracle Wallet	5-17
Creating a Wallet	5-17
Specifying the Location of Wallets.....	5-18
Enabling Wallets to Open on Windows	5-19
Task 7: Specify Caching Rules	5-21
Task 8: Restart Oracle Web Cache	5-21

6 Creating Rules for Cached Content

Cacheability Rules Overview	6-2
Cacheability Rule Creation.....	6-2
Cacheability Rule Syntax.....	6-4
Default Cacheability Rules	6-5
Configuring Cacheability Rules	6-6
Cacheability Rule Example.....	6-12
Configuring Expiration Rules	6-14
Configuring Rules for Multiple-Version Documents Containing Cookies	6-16
Configuring Rules for Multiple-Version Documents Containing HTTP Request Headers	6-18
Configuring Rules for Pages with Simple Personalization	6-19
Example: Personalized Page Configuration	6-23
Configuring Rules for Pages with Session Tracking	6-28
Configuring Pages for Content Assembly and Partial Page Caching	6-33
Enabling Partial Page Caching.....	6-33
Using ESI for Simple Personalization	6-34
Examples of ESI Usage.....	6-35
Example Portal Site Implementation	6-36
Example of Simple Personalization with Variable Expressions	6-44
Configuring Cacheability Attributes in Response Headers	6-45
Usage Notes.....	6-46
Example Usage.....	6-46

7 Configuration Considerations for Web Sites with Multiple Application Web Servers

Configuring Load Balancing and Failover	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
Invalidating Documents in the Cache	8-3
Setting the Invalidation Port Number	8-3
Sending Invalidation Messages	8-5
Manual Invalidation Using Telnet	8-6
Manual Invalidation Using Oracle Web Cache Manager.....	8-12
Automatic Invalidation Using Database Triggers.....	8-16

Automatic Invalidation Using Scripts.....	8-16
Automatic Invalidation Using Applications	8-16
Invalidation Examples	8-17
Example: Invalidating One Document.....	8-17
Example: Invalidating Multiple Objects	8-19
Example: Invalidating a Subtree of Documents	8-20
Example: Invalidating All Documents for a Web Site	8-21
Example: Invalidating Documents with the Prefix	8-22
Evaluating Event Logs	8-23
Format of the Event Log File.....	8-23
Event Log Examples.....	8-23
Example: Event Log with Startup Entries.....	8-23
Example: Event Log with Unsuccessful Startup Entries	8-24
Example: Event Log with an Invalidation Entry	8-24
Example: Event Log with an Invalidation Message Error.....	8-24
Example: Event Log with Shutdown Entries	8-25
Finding Errors in the Event Log.....	8-25
Configuring Event Logs.....	8-25
Evaluating Access Logs.....	8-27
Format of the Access Log File	8-27
Access Log Examples	8-29
Example: Access Log with Reload Entries.....	8-30
Example: Access Log with Wrong Path Entry	8-30
Example: Access Log with Status Code 404 Entry	8-30
Example: Access Log with Status Code 304 Entry	8-30
Configuring Access Logs.....	8-31

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	9-7

10 Troubleshooting Oracle Web Cache Configuration

Startup Failures	10-2
Port Conflicts	10-2
Cache Memory	10-4
Privileged Ports	10-5
Greater Than One Thousand Maximum Connections	10-7
Wallet Cannot Be Opened	10-8
Application Web Server Capacity	10-10
Wrong or Older Cached Content	10-11
Load on Oracle Web Cache Computer	10-11
Configuration Changes Made in Oracle Web Cache Manager	10-12

11 A Case Study Deployment

About Digital River	11-2
Content of Digital River	11-2
Hardware/Network Deployment of Digital River Before Oracle Web Cache	11-2
Why Oracle Web Cache?	11-4
Hardware/Network Deployment with Oracle Web Cache	11-4
Cache Configuration	11-6
Performance Results with Oracle Web Cache	11-7

Part III Reference

A Oracle Web Cache Directory Structure

B Oracle Web Cache Default Settings

C Invalidation Document Type Definition

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

D Edge Side Includes Language

Overview of ESI Tag Library	D-2
Syntax Rules	D-3
Nesting Elements	D-3
Variable Expressions	D-4
Usage	D-5
Variable Substructure Access	D-5
Variable Default Values	D-6
Exceptions and Errors	D-7
Apology Page	D-7
ESI Language Control	D-7
Enabling ESI	D-7
ESI Tag Descriptions	D-8
ESI include Tag	D-9
ESI choose when otherwise Tags	D-10
ESI try attempt except Tags	D-14
ESI comment Tag	D-15
ESI remove Tag	D-16
ESI <!--esi-->Tag	D-17
ESI vars Tag	D-18

E Event Log Messages

Information Events	E-2
Warning Events	E-4
Error Events	E-6

Glossary

Index

List of Figures

1-1	Oracle Web Cache Architecture	1-3
1-2	Web Server Acceleration	1-5
1-3	Load Balancing	1-11
1-4	Failover	1-13
1-5	Application Web Server Binding	1-15
1-6	SSL for Secure Connections	1-16
2-1	Performance Assurance Heuristics Graph	2-6
2-2	Multiple-Version Document	2-9
2-3	Page with a Personalized Attribute	2-13
2-4	Session-Encoded URLs	2-16
2-5	Template Page.....	2-18
2-6	ESI Markup.....	2-19
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	Deploying Oracle Web Cache to Receive HTTP and HTTPS Requests.....	3-5
3-4	Forwarding HTTPS Requests To a Dedicated Oracle Web Cache Server.....	3-6
3-5	Forwarding HTTPS Requests To an Application Web Server	3-8
3-6	Load Balancing with Oracle Web Cache.....	3-9
3-7	Accelerating Portions of a Web Site.....	3-11
3-8	Configuring Multiple Oracle Web Caches as a Failover Pair	3-13
3-9	Configuring Oracle Web Cache Inside a Firewall	3-15
3-10	Configuring Oracle Web Cache Outside a Firewall.....	3-16
3-11	Distributed Caching.....	3-18
3-12	Centralizing the Data Source	3-20
4-1	Oracle Web Cache Manager Interface	4-3
4-2	Cacheability Rules Property Sheet.....	4-6
6-1	Default Cacheability Rules.....	6-5
6-2	Cacheability Rules Example	6-12
6-3	monthly.htm.....	6-23
6-4	Edit/Create Session/Personalized Attribute Definition Dialog Box	6-24
6-5	Add Session/Personalized Attribute Related Caching Rule Dialog Box.....	6-25
6-6	Create Cacheability Rule Dialog Box.....	6-26
6-7	monthly.htm When Cached.....	6-27
6-8	Portal Site Page	6-36
6-9	portal.esi Example: Rotating Banner and Personalized Greeting	6-37
6-10	portal.esi Example: Rotating Banner and Personalized Greeting Output	6-38
6-11	portal.esi Example: Rotating Banner and Personalized Greeting Reload.....	6-38
6-12	portal.esi Example: My Stocks, World Markets, and Movers Sections	6-39
6-13	portal.esi Example: PersonalizedStockSelection Fragment for jesse	6-40
6-14	portal.esi Example: PersonalizedStockSelection Fragment for scott	6-41

6-15	portal.esi Example: Headline News, Weather, and Sports Sections	6-42
6-16	PL/SQL Code without Personalization	6-44
6-17	PL/SQL Code with Personalization through ESI	6-44
8-1	Invalidation	8-5
11-1	Digital River Deployment Before Oracle Web Cache	11-3
11-2	Digital River Deployment with Oracle Web Cache.....	11-5
11-3	Cacheability Rules	11-6
C-1	Invalidation Request DTD.....	C-2
C-2	Invalidation Response DTD	C-5
D-1	Nested ESI Elements	D-3
D-2	Statement Placement	D-12

List of Tables

2-1	Multiple-Version Document with Different Cookie Values	2-10
2-2	HTTP Request-Header Fields	2-11
2-3	Summary of ESI Tags	2-19
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	6-4
6-2	Default Cacheability Rules	6-5
6-3	Regular Expression Examples	6-13
6-4	Surrogate-Control Control Directives	6-45
8-1	Invalidation Message Syntax	8-8
8-2	Invalidation Response Syntax	8-12
8-3	XLF Fields for Access Logs	8-27
9-1	Oracle Web Cache Health Monitor Statistics	9-4
9-2	Oracle Web Cache Statistics	9-5
9-3	Application Web Server Statistics	9-8
11-1	Digital River Cacheability Rules	11-7
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	ESI-Supported Variables	D-4
D-2	Dictionary and List Examples	D-6
E-1	Information Events	E-2
E-2	Warning Events	E-4
E-3	Error Events	E-6

Send Us Your Comments

Oracle9iAS Web Cache Administration and Deployment Guide, Release 2.0.0

Part No. A90372-04

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: Server Technologies Documentation Manager
- Postal service:

Oracle Corporation
Server Technologies Documentation
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

Oracle9iAS 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](#)
- [Documentation Accessibility](#)

Audience

Oracle9iAS 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 guide, 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:

Part I, "Getting Started with Oracle Web Cache"

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.

Part II, "Configuration and Administration of Oracle Web Cache"

Chapter 5, "Initial Setup and Configuration"

This chapter describes the steps to initially configure Oracle Web Cache to begin caching content.

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 or 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 interpret them.

Chapter 10, "Troubleshooting Oracle Web Cache Configuration"

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

Chapter 11, "A Case Study Deployment"

This chapter describes how Digital River, a global Commerce Service Provider (CSP), deployed Oracle Web Cache.

Part III, "Reference"

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 Definition"

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

Appendix D, "Edge Side Includes Language"

This appendix describes the **Edge Side Includes (ESI)** language used for content assembly of dynamic HTML fragments.

Appendix E, "Event Log Messages"

This appendix describes the most common event log messages.

Glossary

Related Documentation

For more information, see the Oracle9i Application Server documentation set, especially:

- *Oracle9i Application Server Overview Guide*
- *Oracle9i Application Server Oracle Wallet Manager User's Guide*
- *Oracle HTTP Server powered by Apache Performance Guide*
- *OracleJSP Support for JavaServer Pages Developer's Guide and Reference*
- Oracle PL/SQL documentation

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://rfc.net/rfc2616.html> for further information about the HTTP protocol
- <http://rfc.net/rfc2965.html> for further information about the Set-Cookie response header
- <http://rfc.net/rfc1421.html> for further information about password base64 encoding
- <http://www.edge-delivery.org> for further information about the **Edge Side Includes (ESI)** language

Conventions

This section describes the conventions used in the text and code examples of this documentation. It describes:

- [Conventions in Text](#)
- [Conventions in Code Examples](#)
- [Conventions for Windows Operating Systems](#)

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.	When you specify this clause, you create an index-organized table .
<i>Italics</i>	Italic typeface indicates book titles or emphasis.	<i>Oracle9i Application Server Overview Guide</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)	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. Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	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) <i>italic</i>	Lowercase monospace italic font represents placeholders or variables.	You can specify the <i>parallel_clause</i> . Run <code>Uold_release.SQL</code> where <i>old_release</i> 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 subquery; SELECT col1, col2, ... , coln 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); 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/system_password DB_NAME = database_name</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; SELECT * FROM USER_TABLES; 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; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;</pre>

Conventions for Windows Operating Systems

The following table describes conventions for Windows operating systems and provides examples of their use.

Convention	Meaning	Example
Choose Start >	How to start a program. For example, to start Oracle Database Configuration Assistant, you must click the Start button on the taskbar and then choose Programs > Oracle - <i>HOME_NAME</i> > Network Administration > Wallet Manager.	Choose Start > Programs > Oracle - <i>HOME_NAME</i> > Network Administration > Wallet Manager
C:\>	Represents the Windows command prompt of the current hard disk drive. Your prompt reflects the subdirectory in which you are working. Referred to as the command prompt in this guide.	C:\webcache\bin
<i>HOME_NAME</i>	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	C:\> net start Oracle <i>HOME_NAME</i> WebCache

Documentation Accessibility

Oracle's goal is to make our products, services, and supporting documentation accessible to the disabled community with good usability. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

<http://www.oracle.com/accessibility/>

JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

What's New in Oracle Web Cache?

The new features for Oracle Web Cache in release 2.0 include:

- **Edge Side Includes (ESI)**

ESI is a simple markup language that enables content assembly of dynamic HTML fragments. It provides for assembly by enabling Web pages to be broken down into fragments of differing cacheability profiles. Each fragment is a separate object with its own cacheability rule.

See Also:

- ["Content Assembly and Partial Page Caching"](#) on page 2-17
- [Appendix D, "Edge Side Includes Language"](#)

- **Secure Sockets Layer (SSL) Support**

In addition to **HTTP protocol** requests, Oracle Web Cache is able to cache pages for **HTTPS protocol** requests.

See Also: ["Secure Sockets Layer \(SSL\) Support"](#) on page 1-16

- **Cacheability Selectors**

In addition to a document's **URL**, cacheability can also be evaluated against a document's **HTTP request method** or the body of an HTTP **POST method**.

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

- **Cacheability Attributes in HTTP Response Messages**

Application developers can add some of the cacheability attributes to the header of an HTTP response message for a document. This feature enables the application Web server to override the settings configured through the Oracle Web Cache Manager interface, as well as allowing other third-party caches to use Oracle Web Cache cacheability attributes.

See Also: ["Configuring Cacheability Attributes in Response Headers"](#) on page 6-45

- **Cache Status Information in HTTP Response Messages**

Cache hit and cache miss information is added to the `Server` response-header field of the HTTP response message. This feature enables you to determine whether a request was served from the cache or the application Web server.

See Also: ["Request and Response Header Fields"](#) on page 2-3

- **Improved Invalidation**

Invalidation messages can be based on the exact URL that includes the complete path and file name or more advanced invalidation selectors. Advanced selectors include the URL prefix, **HTTP request method**, cookie, and **HTTP request header**.

See Also: ["Invalidating Documents in the Cache"](#) on page 8-3

- **On-the-Fly Compression**

In addition to cacheable documents, non-cacheable documents can be now be compressed.

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

Part I

Getting Started with Oracle Web Cache

Part I provides an overview of Oracle Web Cache concepts, products, and tools.

This part contains the following chapters:

- [Chapter 1, "Introduction to Oracle Web Cache"](#)
- [Chapter 2, "Oracle Web Cache Concepts"](#)
- [Chapter 3, "Deploying Oracle Web Cache"](#)
- [Chapter 4, "Configuration and Administration Tools Overview"](#)

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.

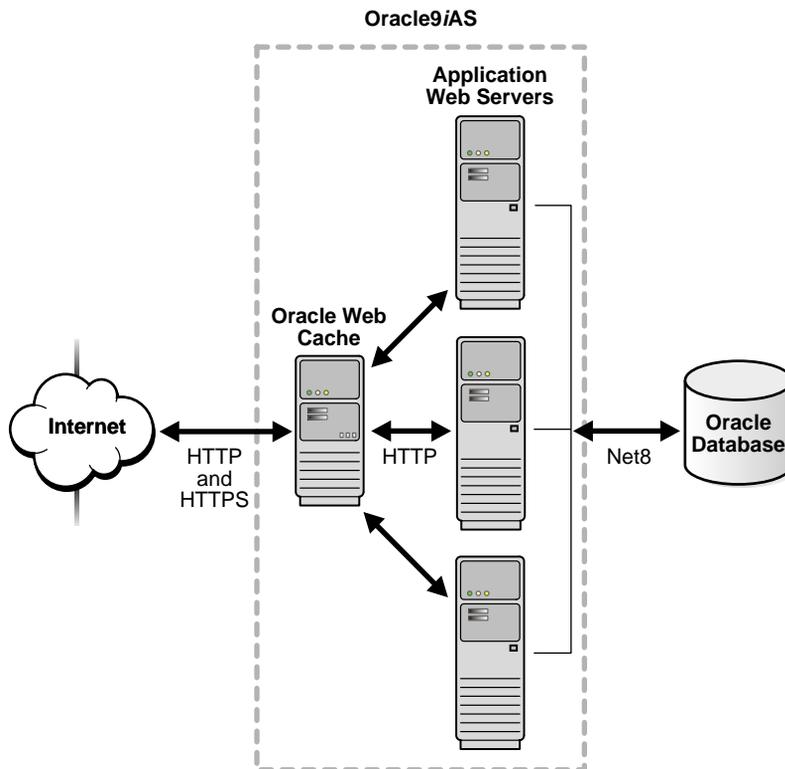
Oracle's Solution to Web Site Performance Issues

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 Oracle9i Application Server and Oracle8i.

By storing frequently accessed URLs in 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. Because Oracle Web Cache is able to cache more content than legacy proxies, it provides optimal performance by greatly reducing 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 protocol** or **HTTPS protocol** 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:

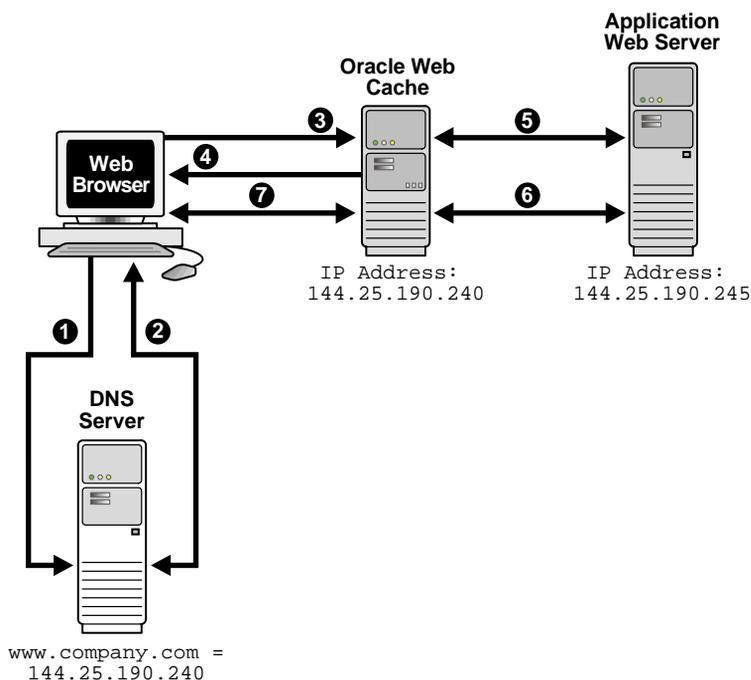
1. A browser sends a request to a Web site named `www.company.com`.
This request in turn generates a request to Domain Name System (DNS) for the IP address of the Web site.
2. DNS returns the IP address of Oracle Web Cache, that is, 144.25.190.240.
3. The browser sends the request for the Web page to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

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 for each 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
- Pages that require content assembly of dynamic fragments

See Also:

- ["Caching Dynamically Generated Content"](#) on page 2-7 for further information about dynamically-generated content
- ["Content Assembly and Partial Page Caching"](#) on page 2-17

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 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 2-4 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 2-4 for further information about performance assurance

Surge Protection of Application Web Servers

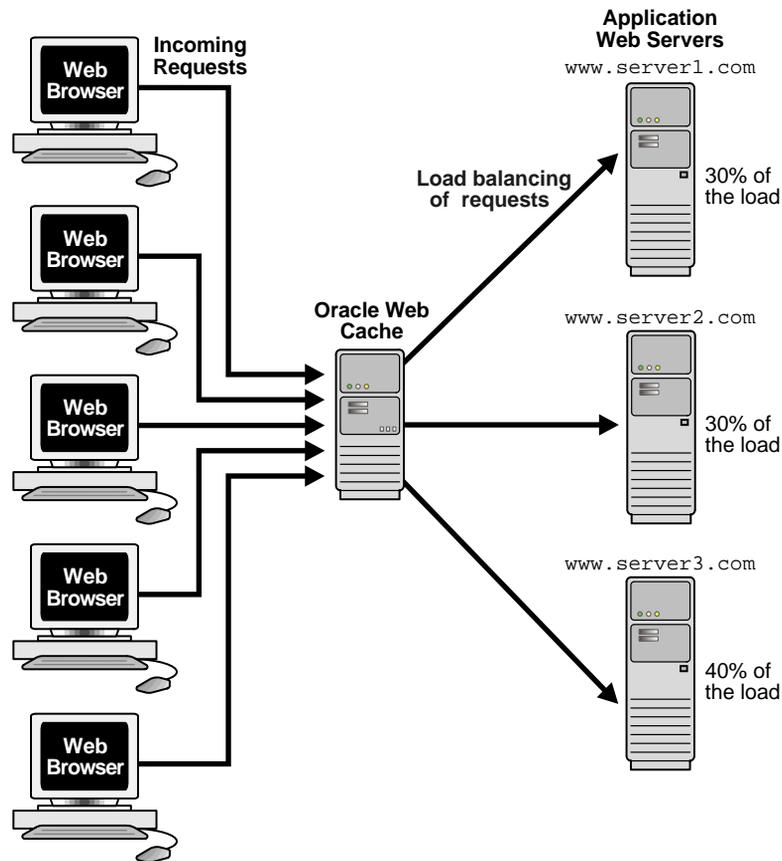
Oracle Web Cache passes requests for non-cacheable or stale documents to the application Web servers. To prevent an overload of requests on the application Web servers, Oracle Web Cache has a surge protection feature that enables you to set a limit on the number of concurrent requests that the application 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 and HTTPS 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 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



See Also:

- "[Task 3: Specify Web Site Settings](#)" on page 5-5 for configuration details
- "[Configuring Load Balancing and Failover](#)" on page 7-3 for configuration details

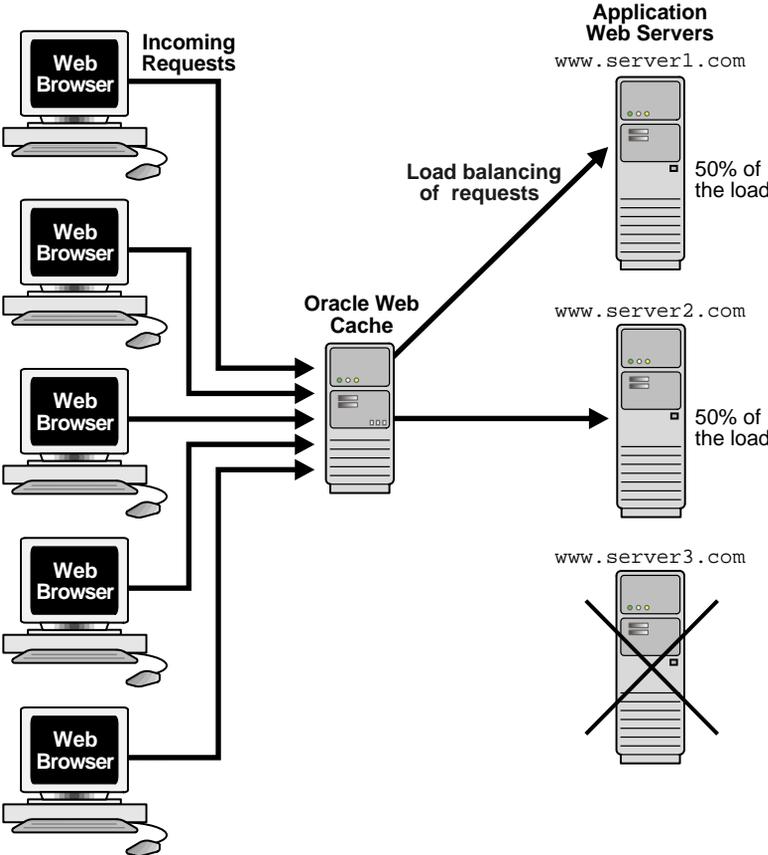
Backend Failover

After a specified number of 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 until it is back online.

The **failover** feature is shown in [Figure 1-4](#) on page 1-13. An outage of `www.server3.com`, which was receiving 40 percents 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



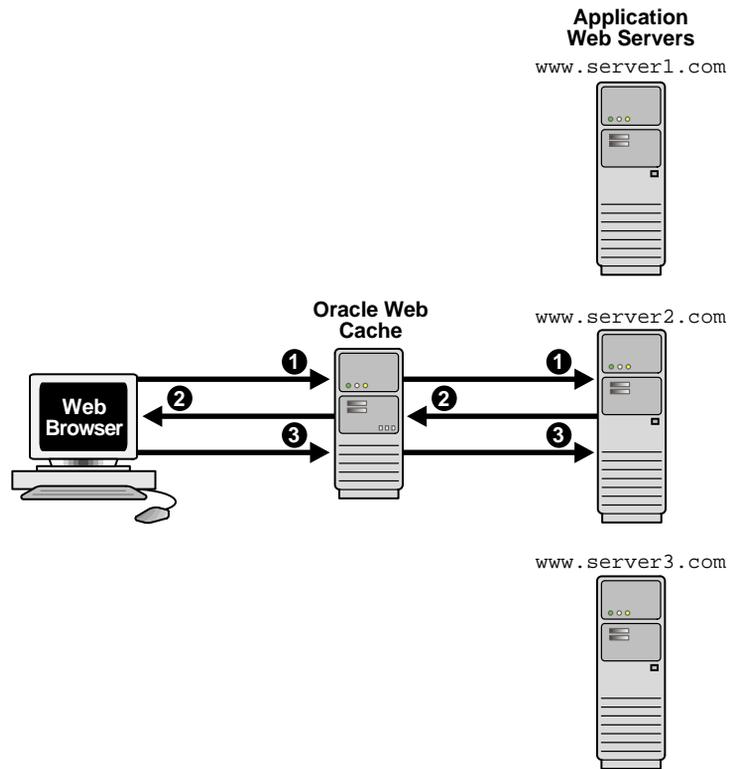
Application Web Server Binding

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](#) on page 1-15 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.

See Also: ["Binding a Session to an Application Web Server"](#) on page 7-3 for configuration details

Figure 1-5 Application Web Server Binding

Security Features

Oracle Web Cache provides the following security-related features:

- [Restricted Administration](#)
- [Secure Sockets Layer \(SSL\) Support](#)

Restricted Administration

Oracle Web Cache restricts administration with the following features:

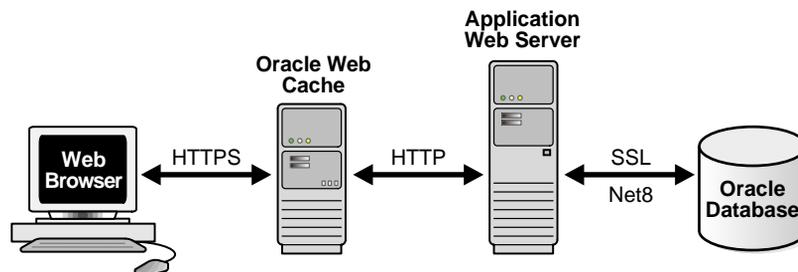
- Password authentication for administration and invalidation operations
- Control over which ports administration and invalidation operations can be requested from
- IP and subnet administration restrictions

Secure Sockets Layer (SSL) Support

The [Secure Sockets Layer \(SSL\)](#) protocol, developed by Netscape Corporation, is an industry-accepted standard for network transport layer security. SSL provides authentication, encryption, and data integrity, in a public-key infrastructure (PKI). By supporting SSL, Oracle Web Cache is able to cache pages for HTTPS requests.

As shown in [Figure 1-6](#), you can configure Oracle Web Cache to receive HTTPS browser requests. In addition, you can secure connections between the application Web server and the database with SSL.

Figure 1-6 SSL for Secure Connections



Limitations: HTTPS support has the following limitations in this release:

- Oracle Web Cache does not provide authentication or access control.
 - Oracle Web Cache does not support client-side certification.
 - The HTTPS protocol is not supported between Oracle Web Cache and application Web server(s).
-
-

SSL interacts with the following entities:

- [Certificate Authority](#)
- [Certificate](#)
- [Wallet](#)

Certificate Authority A certificate authority (CA) is a trusted third party that certifies the identity of third parties and other entities, such as users, databases, administrators, clients, and servers. The certificate authority verifies the party identity and grants a certificate, signing it with the its private key. The Oracle Web Cache certificate must be signed by a CA.

Different CAs may have different identification requirements when issuing certificates. One may require the presentation of a user's driver's license, while others may require notarization of the certificate request form, or fingerprints of the requesting party.

The CA publishes its own certificate, which includes its public key. Each network entity has a list of certificates of the CAs it trusts. Before communicating with another entity, a given entity uses this list to verify that the signature on the other entity's certificate is from a known, trusted CA.

Network entities can obtain their certificates from the same or different CAs. By default, Oracle Advanced Security automatically installs trusted certificates from VeriSign, RSA, Entrust, and GTE CyberTrust when you install a new wallet (See: [Wallet](#)).

Certificate A certificate is created when a party's public key is signed by a trusted CA. A certificate ensures that a party's identification information is correct, and that the public key actually belongs to that party.

A certificate contains the party's name, public key, and an expiration date—as well as a serial number and certificate chain information. It can also contain information about the privileges associated with the certificate.

When a network entity receives a certificate, it verifies that it is a trusted certificate—one issued and signed by a trusted certificate authority. A certificate remains valid until it expires or is terminated.

Wallet A wallet is a transparent database used to manage authentication data such as keys, certificates, and trusted certificates needed by SSL. A wallet has an X.509 version 3 certificate, private key, and list of trusted certificates.

Security administrators use the Oracle Wallet Manager to manage security credentials on the Oracle Web Cache server. Wallet owners use it to manage security credentials on clients. Specifically, Oracle Wallet Manager is used to do the following:

- Generate a public-private key pair and create a certificate request for submission to a certificate authority.
- Install a certificate for the identity.
- Configure trusted certificates for the identity.

Note: Installation of Oracle9iAS Release 1.0.2.2 also installs Oracle Wallet Manager release 2.0.

See Also: *Oracle Wallet Manager User's Guide*

How SSL Works The authentication process between the browser and Oracle Web Cache consists of the following basic steps:

1. The browser initiates a connection to Oracle Web Cache by using HTTPS.
2. SSL performs the handshake between the browser and Oracle Web Cache.

At the commencement of an HTTPS network connection between a browser and Oracle Web Cache, an SSL handshake is performed. An SSL handshake between the browser and Oracle Web Cache includes the following actions:

- The browser and Oracle Web Cache establish which cipher suites to use.
- Oracle Web Cache sends its certificate to the browser, and the browser verifies that the Oracle Web Cache's certificate was signed by a trusted CA.
- The browser and Oracle Web Cache exchange key information using public key cryptography; based on this information, each generates a session key. All subsequent communications between the browser and the Oracle Web Cache is encrypted and decrypted by using this set of session keys and the negotiated cipher suite.

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 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 Oracle Web Cache compress both cacheable and non-cacheable documents upon insertion into the cache for browsers. Because compressed documents are smaller in size, they are delivered faster to browsers with fewer round-trips, reducing overall **latency**. On average, Oracle Web Cache is able to compress text files by a factor of 4. For example, 300 KB files are compressed down to 75 KB.

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](#)
- [Request and Response Header Fields](#)
- [Cache Freshness and Performance Assurance](#)
- [Caching Dynamically Generated Content](#)
- [Content Assembly and Partial Page Caching](#)

Populating Oracle Web Cache

Oracle Web Cache uses cacheability rules to determine which documents to cache. When cacheability rules for a particular URL are first configured, those documents contained within the URL are not cached until there is a browser request for them. When the first request for a document comes in, Oracle Web Cache appends a `Surrogate-Capability` request-header field to the document. The `Surrogate-Capability` request-header field identifies that the document passed through the cache. Oracle Web Cache then sends the request to the application Web server. This is a **cache miss**. If the requested document is specified as one of the documents to cache, then Oracle Web Cache caches the document for subsequent requests. For a subsequent request for the document, Oracle Web Cache serves the document from its cache to the browser. This is a **cache hit**.

When a browser sends a **GET method** request with an `If-Modified-Since` request-header field for a cached document, Oracle Web Cache compares the time stamp used in header with the `Last-Modified` request-header field of the cached document to determine if the document needs to be served. If the cached document is more current than the one requested by the browser, then Oracle Web Cache serves the cached document to the browser. When the `Last-Modified` header does not exist, Oracle Web Cache uses the time the document entered the cache as the time stamp. If the cached document is less current than the one requested by the browser, then Oracle Web Cache sends a 304 status code to the browser.

If a document contains a **cookie**, then 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, then Oracle Web Cache caches the response. Because a session value change does not necessarily indicate a change of state on the application Web servers, **session cookie** values are not evaluated. For documents that use these cookies, the response is cached, regardless of whether or not the cookie values match.

Notes:

- You can populate the cache with the aid of the Apache Benchmark tool. See the Apache documentation for further information.
 - When you stop Oracle Web Cache, all objects are cleared from the cache. In addition, all statistics are cleared.
-
-

See Also: ["Caching Dynamically Generated Content"](#) on page 2-7 for an overview of cookies

Request and Response Header Fields

For each requested document from the cache, Oracle Web Cache appends a `Surrogate-Capability` request-header field to a document's HTTP request message. The `Surrogate-Capability` request-header enables Oracle Web Cache to identify the operations it is capable of performing to application Web servers. The `Surrogate-Capability` request-header field has the following syntax:

```
Surrogate-Capability: orcl="operation_value"
```

where `"operation_value"` is one of the following:

- `"ESI/1.0"` to process ESI release 1.0 tags for content assembly and **partial page caching**
- `"WEBCACHETAG"` to process the `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags for **personalized attributes** and **session-encoded URLs**

Note: `"WEBCACHETAG"` is mutually exclusive with `"ESI/1.0"`.

For documents sent to browsers, Oracle Web Cache adds cache hit or cache miss information to the `Server` response-header field of the HTTP response message:

```
Server: Oracle9iAS Web Cache/release M|H|S /Apache/release  
(operating_system)
```

where:

- M indicates a cache miss
- H indicates a cache hit
- S indicates a stale hit

In the following example, the `Server` field specifies that the document was a cache miss:

```
Server: Oracle9iAS Web Cache/2.0.0.2.0 M /Apache/1.3.12 (Unix)  
(Red Hat/Linux)
```

Using this information, you can determine whether a request was served from the cache or the application Web server.

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 application Web servers. With invalidation, an 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 removed and then refreshed with new content from the application Web servers. You can select to remove and refresh invalid documents immediately, or base the removal and refresh on the current load of the application Web servers.

Expirations are useful if it can be accurately predicated when content will change on an application Web server or database. An invalidation message 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 to browsers. For this reason, Oracle Web Cache intelligently serves some of the documents stale until the application Web servers have the capacity to refresh them.

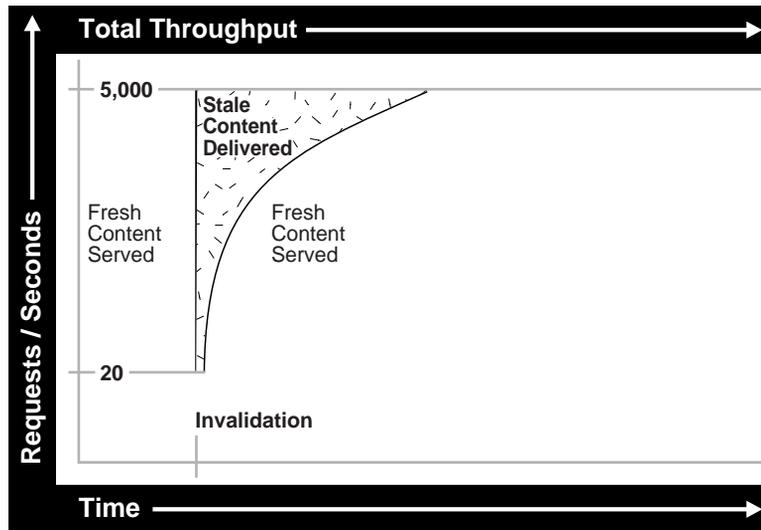
Oracle Web Cache provides minimal trade-off between performance and consistency through **performance assurance heuristics** that determine which documents can be served stale. These heuristics are based on a number of factors including:

Validity	<p>Validity is based on the expiration time, invalidation time, and removal time of an object.</p> <p>Oracle Web Cache calculates validity by comparing the current time relative to an object's expiration/invalidation time and the object's scheduled removal time. Prior to expiration/invalidation time, the object is considered valid. Between expiration/invalidation time and removal time, the object's validity level decreases linearly. During this interim state, objects with a higher validity level have a higher propensity to be served stale. When current time reaches removal time, the object is considered totally invalid and can no longer be served stale. Scheduled removal time is something that administrators can control. When expiring/invalidating content, administrators have the option to remove objects immediately, which may be necessary for sensitive objects that should never be served stale. Likewise, where some degree of inconsistency is tolerable, administrators can specify a removal time in the near future.</p>
Popularity	<p>Popularity is determined by:</p> <ul style="list-style-type: none">■ The number of times the object has been requested since insertion into the cache■ The number of recent requests for the object
Load of the Application Web Server	<p>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.</p>
Limit on the Application Web Server	<p>The configured limit on the application Web server load is the configured number of concurrent connections the application Web server can safely handle.</p>

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

Figure 2–1 illustrates how performance assurance heuristics are used during widespread invalidation.

Figure 2–1 Performance Assurance Heuristics Graph



Right after invalidation, the number of fresh documents served decreases to 20 documents for each second. However, the number of fresh cache hits quickly increases over a short amount of time. 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 be revalidated in a given period of time is dependent on application Web server capacity. At the end of invalidation, only fresh content is served.

Note: Performance assurance heuristics do not apply when you configure documents to be removed and refreshed immediately.

Caching Dynamically Generated Content

Most Web pages today are dynamically generated before delivery to the browser. Web developers frequently use database-driven technologies like Java Server Pages (JSP), Active Server Pages (ASP), PL/SQL Server Pages (PSP), Java Servlets, and Common Gateway Interface (CGI) to design their applications. These technologies are used for complex Web sites, as they are easier to modify and maintain when information is stored in a database. 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

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.

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 accepts requests that use the following characters as delimiters for **embedded URL parameters**: ampersand (&), dollar sign (\$), or semi-colon (;).

Note: Examples in this guide use ampersand (&) as the delimiter.

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 Document](#)
- [Personalized Attributes](#)
- [Session Information](#)

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

Multiple Versions of the Same Document

Some pages have multiple versions, enabling categorization. [Figure 2-2](#) on page 2-9 shows the same document,

`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 Document



Unit Price

Oracle Store Database CD Packs

Product	Unit Price	# of Units	Buy
Oracle Internet Developer Suite Rel 1.0 CD Pack for MS Windows NT	\$ 39.95		Buy
Oracle Database 8i Release 2 (8.1.6) CD Pack v5 for Aix-Based Systems	\$ 39.95		Buy
Oracle Database 8i Release 2 (8.1.6) CD Pack v5 for Compaq Tru64 UNIX	\$ 39.95		Buy
Oracle Database 8i Release 2 (8.1.6) CD Pack v5 for HP 9000 Series HP-UX	\$ 39.95		Buy
Oracle Database 8i Release 2 (8.1.6) CD Pack v6 for Linux Intel	\$ 39.95		Buy
Oracle Database 8i Release 2 (8.1.6) CD Pack v6 for MS Windows NT	\$ 39.95		Buy
Oracle Database 8i Release 2 (8.1.6) CD Pack v7 for Sun SPARC Solaris	\$ 39.95		Buy
Oracle Database 8i (8.1.5) CD Pack v7 for Aix-Based Systems	\$ 39.95		Buy
Oracle Database 8i (8.1.5) CD Pack v7 for Compaq Tru64 UNIX	\$ 39.95		Buy
Oracle Database 8i (8.1.5) CD Pack v8 for HP 9000 Series HP-UX	\$ 39.95		Buy
Oracle Database 8i (8.1.5) CD Pack v6 for Linux Intel	\$ 39.95		Buy
Oracle Database 8i (8.1.5) CD Pack v9 for MS Windows NT	\$ 39.95		Buy



Unit Price

Oracle Store Database CD Packs

Product	Unit Price	# of Units	Buy
Oracle Internet Developer Suite Rel 1.0 CD Pack for MS Windows NT	\$ 0.00		Buy
Oracle Database 8i Release 2 (8.1.6) CD Pack v5 for Aix-Based Systems	\$ 0.00		Buy
Oracle Database 8i Release 2 (8.1.6) CD Pack v5 for Compaq Tru64 UNIX	\$ 0.00		Buy
Oracle Database 8i Release 2 (8.1.6) CD Pack v5 for HP 9000 Series HP-UX	\$ 0.00		Buy
Oracle Database 8i Release 2 (8.1.6) CD Pack v6 for Linux Intel	\$ 0.00		Buy
Oracle Database 8i Release 2 (8.1.6) CD Pack v6 for MS Windows NT	\$ 0.00		Buy
Oracle Database 8i Release 2 (8.1.6) CD Pack v7 for Sun SPARC Solaris	\$ 0.00		Buy
Oracle Database 8i (8.1.5) CD Pack v7 for Aix-Based Systems	\$ 0.00		Buy
Oracle Database 8i (8.1.5) CD Pack v7 for Compaq Tru64 UNIX	\$ 0.00		Buy
Oracle Database 8i (8.1.5) CD Pack v8 for HP 9000 Series HP-UX	\$ 0.00		Buy
Oracle Database 8i (8.1.5) CD Pack v6 for Linux Intel	\$ 0.00		Buy
Oracle Database 8i (8.1.5) CD Pack v9 for MS Windows NT	\$ 0.00		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 documents 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 document 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 response 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 document 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 Document 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 documents 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-header fields supported.

Table 2–2 HTTP Request-Header Fields

Header Field	Description
Accept	Specifies which media types are acceptable for the response Example: Accept: image/gif, image/x-xbitmap, 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 an HTTP request-header field of User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0) and another request sends an HTTP request-header field 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 like "Hello, *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 `<!-- WEBCACHEEND-->` that Oracle Web Cache can process.

Oracle Web Cache processes 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 2-13 shows two users, Jane Doe and John Doe, accessing the same page,

`http://store.oracle.com/cec/cstage?eccookie=&ecaction=ecpasst`
`hru2&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
<!-- WEBCACHEEND-->
</b>
```

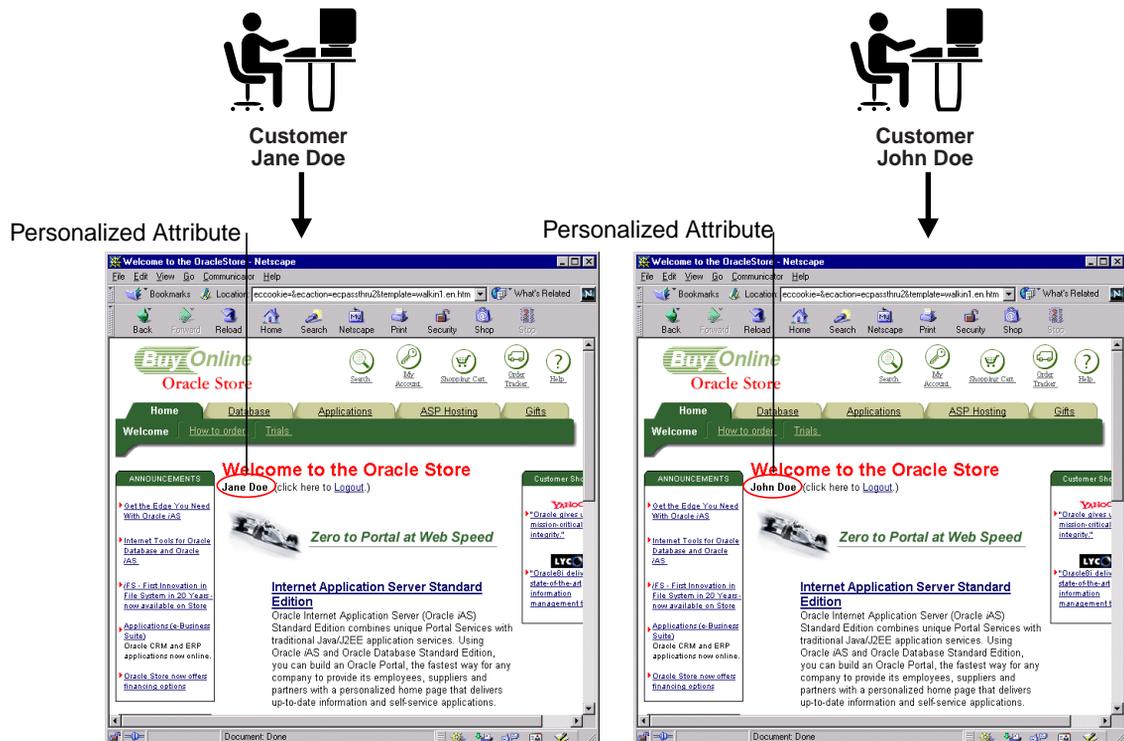
The HTML code for personalized greeting **John Doe** uses the following HTML code:

```
<b>
<!-- WEBCACHETAG="person01"-->
John Doe
<!-- WEBCACHEEND-->
</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 a default 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 Pages with Simple Personalization"](#) on page 6-19

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 that use session cookies or embedded URL parameters, the 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 2-13. 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 6-28

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 2-14, 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](#) on page 2-16 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:

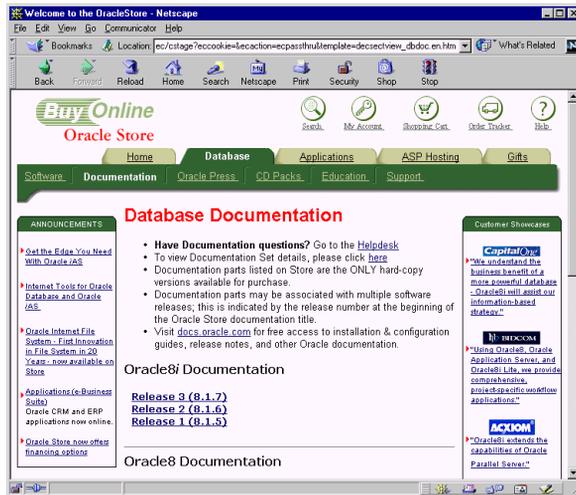
```
<A HREF="/cec/cstage?ecaction=ecproditemlistbysupersect&
ecsid=20330&eccookie=&template=decsectview_pub.en.htm&session_
ID=334326">Release 3 (8.1.7)</A>
```

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 a default 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 Pages with Simple Personalization" on page 6-19

Content Assembly and Partial Page Caching

Oracle Web Cache provides dynamic assembly of Web pages with both cacheable and non-cacheable page fragments. It provides for assembly by enabling Web pages to be broken down into fragments of differing cacheability profiles. These fragments are each maintained as separate elements in the application Web server or content delivery network. The fragments are assembled into HTML pages as appropriate when requested by end users.

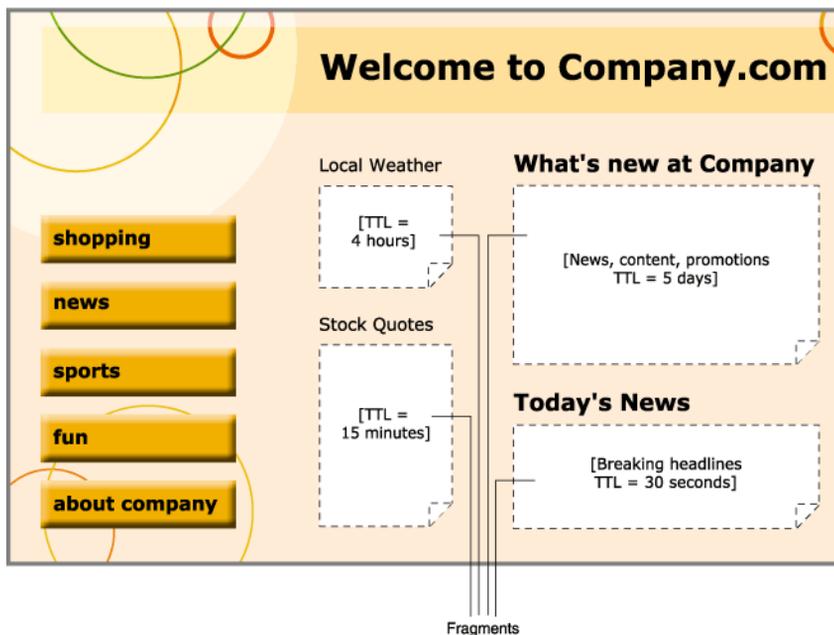
By enabling dynamic assembly of Web pages on Oracle Web Cache rather than on the application Web servers, you can choose to cache some of the fragments of assembled pages. This means that much more HTML content can be cached, then assembled and delivered by Oracle Web Cache when requested. Furthermore, page assembly can be conditional, based on information provided in HTTP request headers or end-user cookies.

- [Page Assembly Components](#)
- [ESI Features](#)
- [ESI for Java \(JESI\)](#)

Page Assembly Components

The basic structure a content provider uses to create dynamic content is a template page containing HTML fragments. As depicted in [Figure 2-5](#), the template consists of common elements, such as a logo, navigation bars, framework, and other "look and feel" elements of the page. The HTML fragments represent dynamic subsections of the page.

Figure 2-5 Template Page



The template page is associated with the URL that end users request. To include the HTML fragments, the template page is configured with [Edge Side Includes \(ESI\)](#) markup tags that tell Oracle Web Cache to fetch and include the HTML fragments. The fragments themselves are HTML files containing discrete text or other objects.

Each included fragment is a separate object with its own cacheability rule. Content providers may want to cache the template for several days, but only cache a particular fragment, such as an advertisement or stock quote, for a matter of seconds or minutes. Other fragments (such as a user's bank account total) may be declared non-cacheable.

Table 2-3 provides a summary of the main ESI tags.

Table 2-3 Summary of ESI Tags

Tag	Description
<code><esi:include></code>	Includes a HTML fragment
<code><esi:choose></code>	Performs conditional processing based on boolean expressions
<code><esi:try></code>	Specifies alternate processing when a request fails because the application Web server is not accessible
<code><esi:vars></code>	Permits variable substitution for environment variables
<code><esi:remove></code>	Specifies non-ESI markup if ESI processing is not enabled
<code><!--esi...--></code>	Specifies content to be processed

Figure 2-6 shows the ESI markup language for the template page shown in Figure 2-5 on page 2-18.

Figure 2-6 ESI Markup

```

<html>
<head>
<title>
Company.com
</title>
</head>
<body>
...
<!-- The following HTML comment tag with an immediate following 'esi' is a
special ESI tag that is removed if and only if this page is processed by an ESI
processor. -->
<!--esi

<esi:comment text="This is the HTML source when ESI is enabled." />

<esi:comment text="Start: The quick link section. You cannot use the standard
HTML comments because the end of that comment tag would disrupt the HTML comment
tag with 'esi' following the two '-'. " />

<esi:comment text="The URI query string parameter 'sessionID' is used to carry
session identifiers, The session ID is encoded in all links. 'type' is used to
categorize this user."/>

```

```
<esi:vars>
  <a href="/shopping.jsp?sessionID=$(QUERY_STRING{sessionID})&type=$(QUERY_
STRING{type})">
    
  </a>
  <a href="/news.jsp?sessionID=$(QUERY_STRING{sessionID})&type=$(QUERY_
STRING{type})">
    
  </a>
  <a href="/sports.jsp?sessionID=$(QUERY_STRING{sessionID})&type=$(QUERY_
STRING{type})">
    
  </a>
  <a href="/fun.jsp?sessionID=$(QUERY_STRING{sessionID})&type=$(QUERY_
STRING{type})">
    
  </a>
  <a href="/about.jsp?sessionID=$(QUERY_STRING{sessionID})&type=$(QUERY_
STRING{type})">
    
  </a>
</esi:vars>

<esi:comment text="End: The quick link section" />
...
<h3>Local Weather</h3>
<esi:include src="/weather.jsp?sessionID=$(QUERY_
STRING{sessionID})&type=$(QUERY_STRING{type})" />
...

<h3>Stock Quotes</h3>
<esi:try>
  <esi:attempt>
    <esi:include src="/CompanyStack.jsp?sessionID=$(QUERY_
STRING{sessionID})&type=$(QUERY_STRING{type})" />
  </esi:attempt>
  <esi:except>
    The company stock quote is temporarily unavailable.
  </esi:except>
</esi:try>
...
```

```

<h3>What's New at Company</h3>
<!-- This section is a static file that does not carry session information -->
<esi:include src="/whatisNew.html" />

...

<h3>Today's News</h3>
<esi:choose>

    <esi:when test="$(QUERY_STRING{type}) == 'Sport'">
        <h4>Sport News</h4>
        <esi:include src="/SportNews.jsp?sessionID=$(QUERY_
STRING{sessionID})&type=$(QUERY_STRING{type})" />
    </esi:when>

    <esi:when test="$(QUERY_STRING{type}) == 'Career'">
        <h4>Financial News</h4>
        <esi:include src="/FinancialNews.jsp?sessionID=$(QUERY_
STRING{sessionID})&type=$(QUERY_STRING{type})" />
    </esi:when>

    <esi:otherwise>
        <h4>General News</h4>
        <esi:include src="/DefaultNews.jsp?sessionID=$(QUERY_
STRING{sessionID})&type=$(QUERY_STRING{type})" />
    </esi:otherwise>

</esi:choose>

...

-->

<!-- This is the HTML source when ESI is disabled. -->
<esi:remove>
Alternative HTML source that does not use ESI goes here. This tag enables you
disable ESI on the fly without redeveloping or redeploying a different home
page.
</esi:remove>

</body>
</html>

```

ESI Features

ESI can be used with HTML, XML, and any Web programming technology. The ESI language includes the following features:

- **Inclusion**

An ESI processor assembles fragments of dynamic content, retrieved from the network, into aggregate pages to output to the user. Each fragment can have its own cacheability rules.
- **Support of variables**

ESI supports the use of variables based on HTTP request attributes. Variables can be used by ESI statements during processing or can be output directly into the processed markup.
- **Conditional processing**

ESI allows use of boolean comparisons for conditional logic in determining how pages are processed.
- **Error handling and alternative processing**

Some ESI tags support specification of a default resource and/or an alternative resource, such as an alternate Web page, if the primary resource cannot be found. Further, it provides an explicit exception-handling statement block.

ESI for Java (JESI)

Edge Side Includes for Java (JESI) is a specification and custom JSP tag library that developers can use to automatically generate ESI code using JSP syntax. Even though JSP developers can always use ESI, JESI provides an even easier way for JSP developers to express the modularity of pages and the cacheability of those modules, without requiring developers to learn a new syntax.

See Also:

- ["Configuring Pages for Content Assembly and Partial Page Caching"](#) on page 6-33
- [Appendix D, "Edge Side Includes Language"](#) and <http://www.edge-delivery.org> for an overview of the ESI language
- *OracleJSP Support for JavaServer Pages Developer's Guide and Reference* for a description of JESI

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:

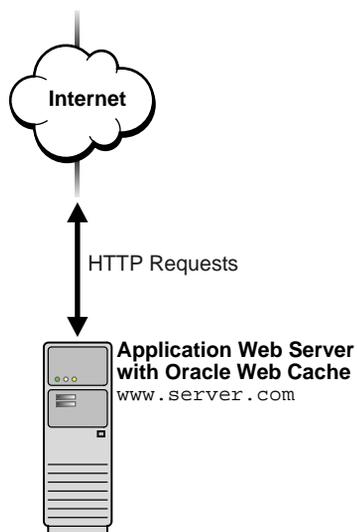
- [Caching Content for One Application Web Server](#)
- [Caching Content for HTTPS Requests](#)
- [Load Balancing Requests Among Application Web Servers](#)
- [Accelerating Portions of a Web Site](#)
- [Using Oracle Web Cache Servers in a Failover Pair](#)
- [Working with Firewalls](#)
- [Deploying Oracle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

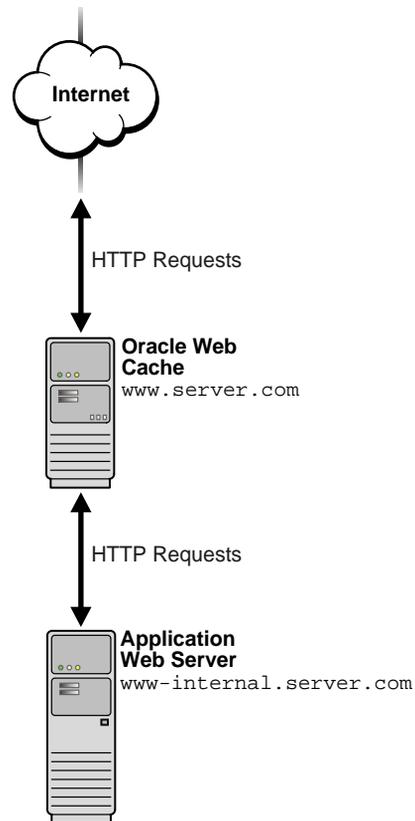
Figure 3-1 Oracle Web Cache On Same Computer As the Application Web Server



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`.

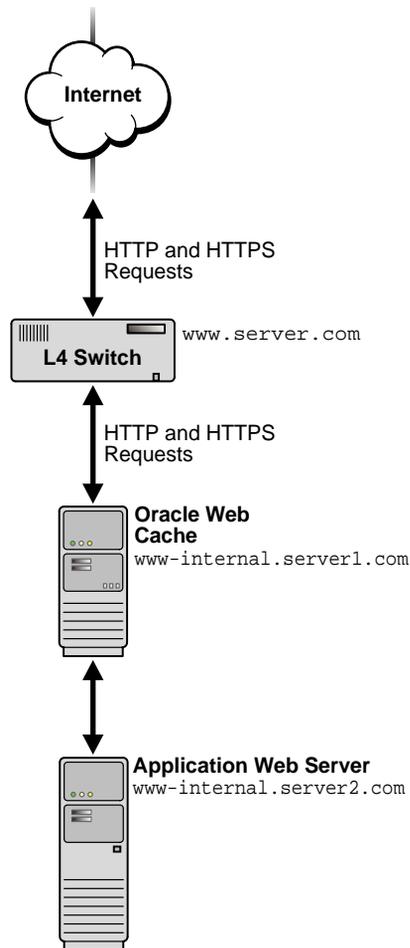
3. Configure Oracle Web Cache with the host name of the application Web server.

Caching Content for HTTPS Requests

In addition to [HTTP protocol](#) requests, you can configure Oracle Web Cache to cache documents for [HTTPS protocol](#) requests. HTTPS requests are typically for secure pages. For an environment with cacheable HTTP and HTTPS requests, you can configure Oracle Web Cache to listen for incoming requests on two ports, one for [Secure Sockets Layer \(SSL\)](#) requests and one for non-SSL requests. In addition, you can use a [Layer 4 \(L4\) switch](#) to pass requests to the appropriate listening port. 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.

[Figure 3-3](#) on page 3-5 shows an L4 switch passing both HTTP and HTTPS requests to Oracle Web Cache server `www-internal.server1.com`.

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

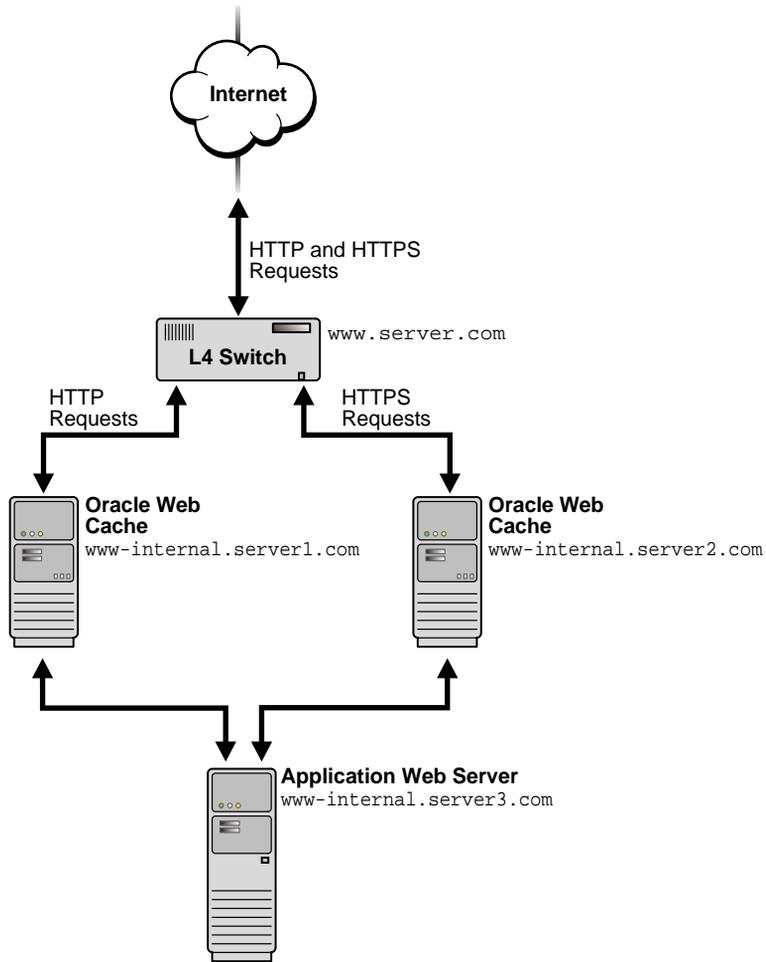
Figure 3-3 Deploying Oracle Web Cache to Receive HTTP and HTTPS Requests

To configure this deployment:

1. Configure the L4 switch with the host name of the Oracle Web Cache server.
2. Configure the Oracle Web Cache server with the host name of the application Web server.
3. Configure the Oracle Web Cache server with an HTTPS listening port.

Figure 3-4 shows two Oracle Web Cache servers receiving requests. HTTP requests are served from server `www-internal.server1.com` and HTTPS requests are served from server `www-internal.server2.com`.

Figure 3-4 Forwarding HTTPS Requests To a Dedicated Oracle Web Cache Server

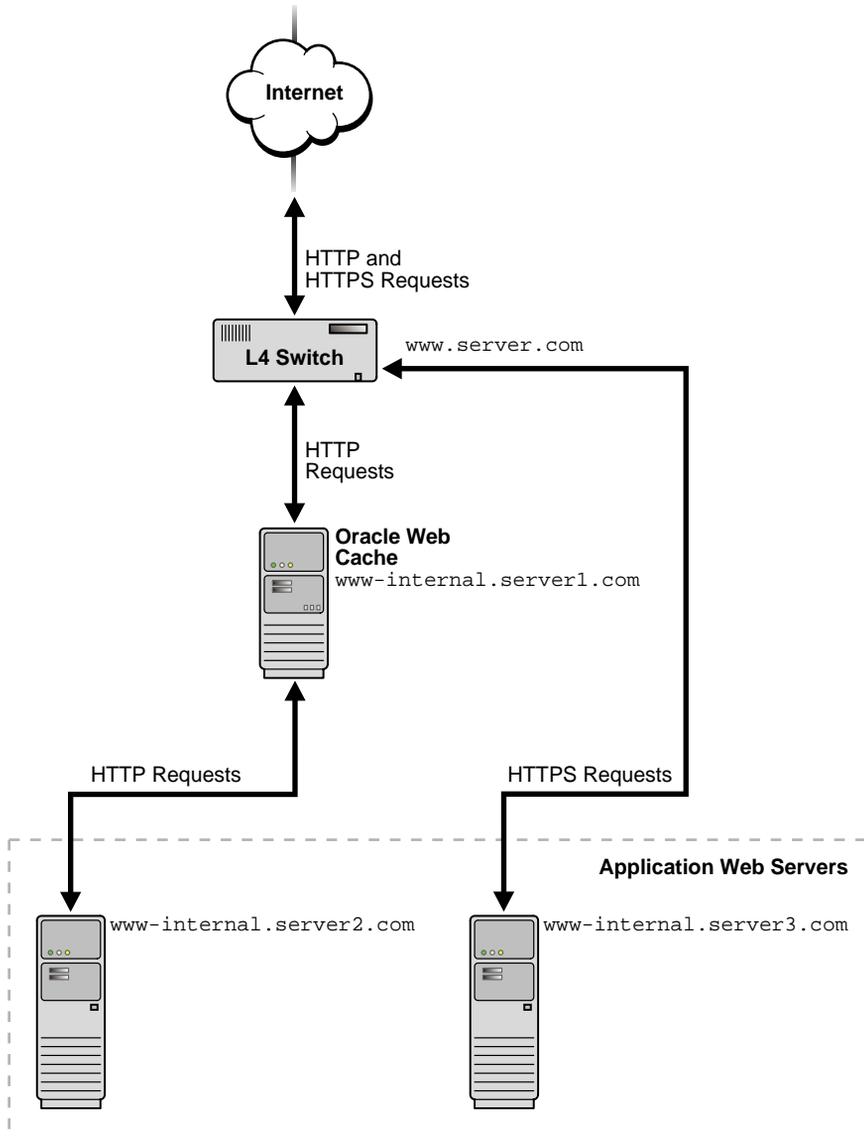


To configure this deployment:

1. Configure the L4 switch with the host names of the Oracle Web Cache servers.
2. Configure the Oracle Web Cache servers with the host name of the application Web server.
3. Configure one of the Oracle Web Cache servers with an HTTP listening port, and configure the other Oracle Web Cache server with an HTTPS listening port.

For many applications, HTTPS is required for secure transactions that should be cached. For example, purchasing pages on an e-commerce site that require credit card information should not be cached. For this type of Web site, you can use an L4 switch to pass all HTTP requests, typically port 80 traffic, to Oracle Web Cache, and forward HTTPS requests for secure pages to a particular application Web server. [Figure 3-5](#) on page 3-8 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`. Note that HTTPS requests could also be passed to `www-internal.server2.com`.

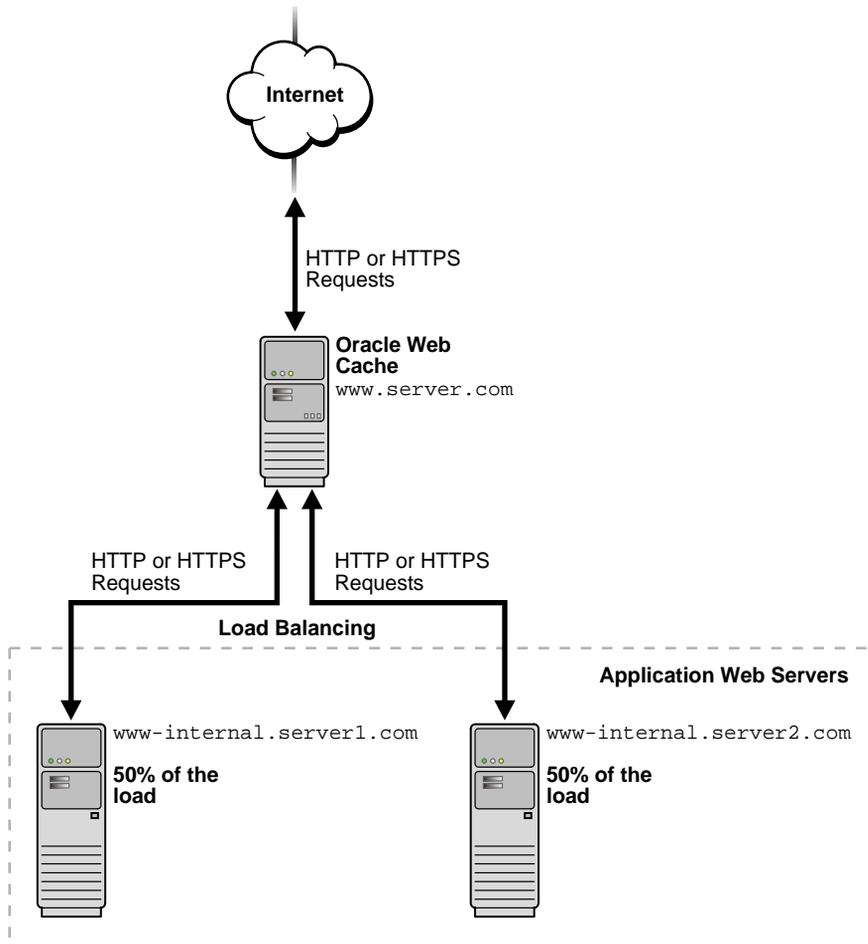
Figure 3-5 Forwarding HTTPS Requests To an Application Web Server



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 3-6](#), you can use Oracle Web Cache to distribute HTTP and HTTPS requests among two or more application Web servers.

Figure 3-6 Load Balancing with Oracle Web Cache



To configure this deployment:

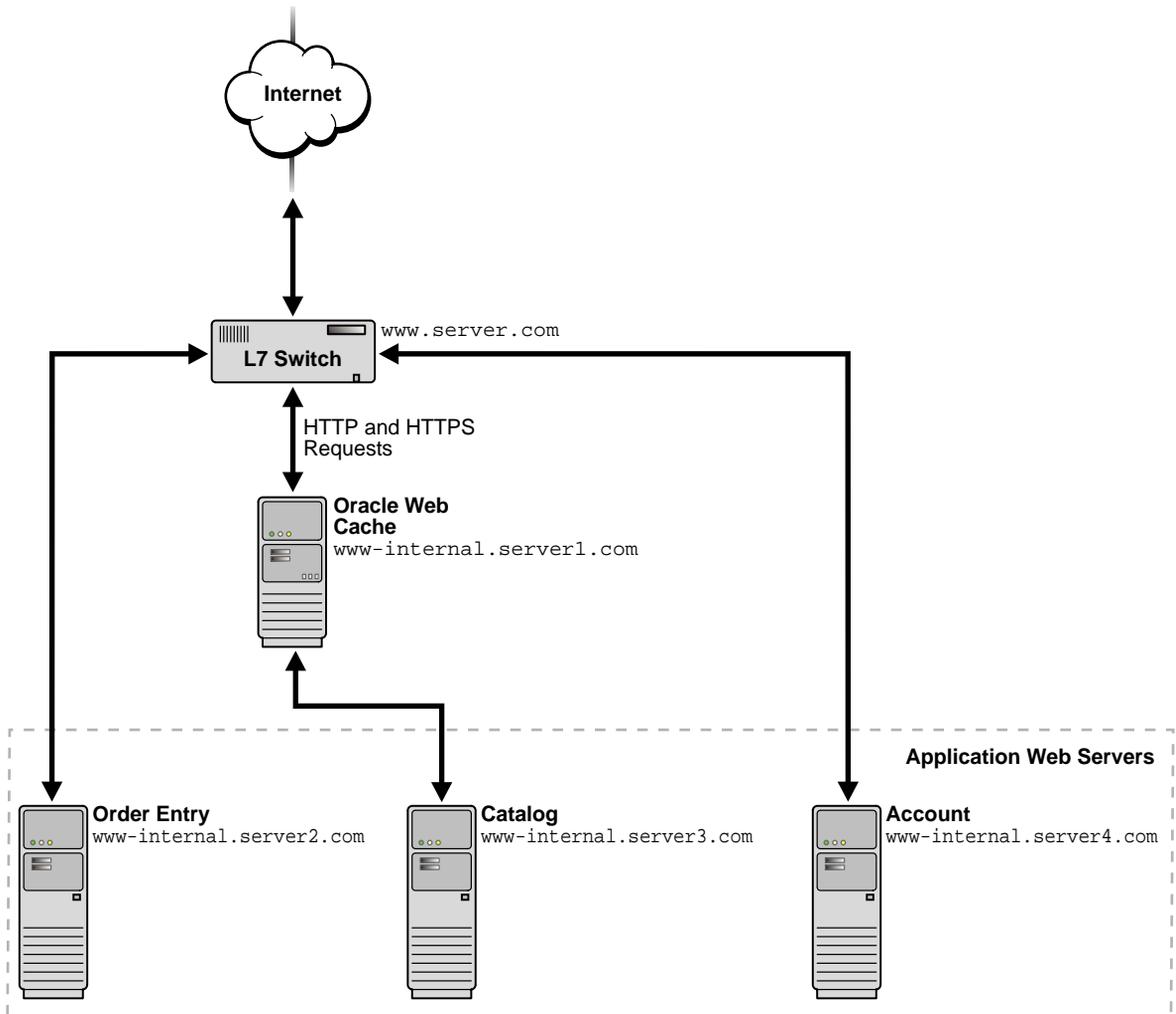
1. Assign the name of the Load Balancer to Oracle Web Cache.
2. Configure Oracle Web Cache with the host names of the application Web servers.

Accelerating Portions of a Web Site

Many Web sites contain cacheable catalog content and non cacheable 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 3-7](#) on page 3-11 shows a **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 3-7 Accelerating Portions of a Web Site

To configure this deployment:

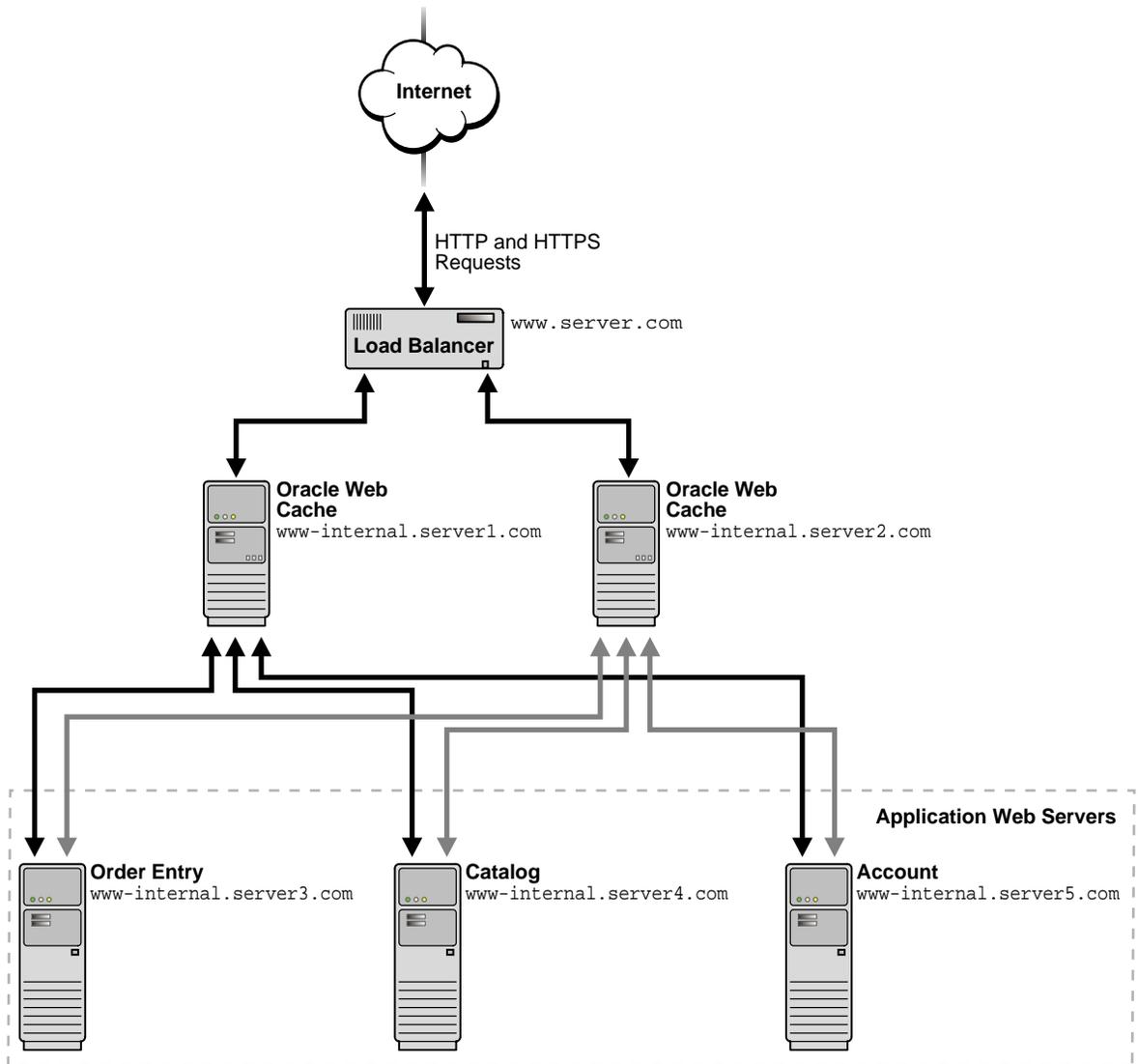
1. Configure the L7 switch with the host name of the Oracle Web Cache server.
2. Configure Oracle Web Cache with the host names of the application Web servers for which it is caching documents.

In [Figure 3-7](#), 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 3-8](#) on page 3-13.

Figure 3–8 *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 3-9](#) on page 3-15 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.

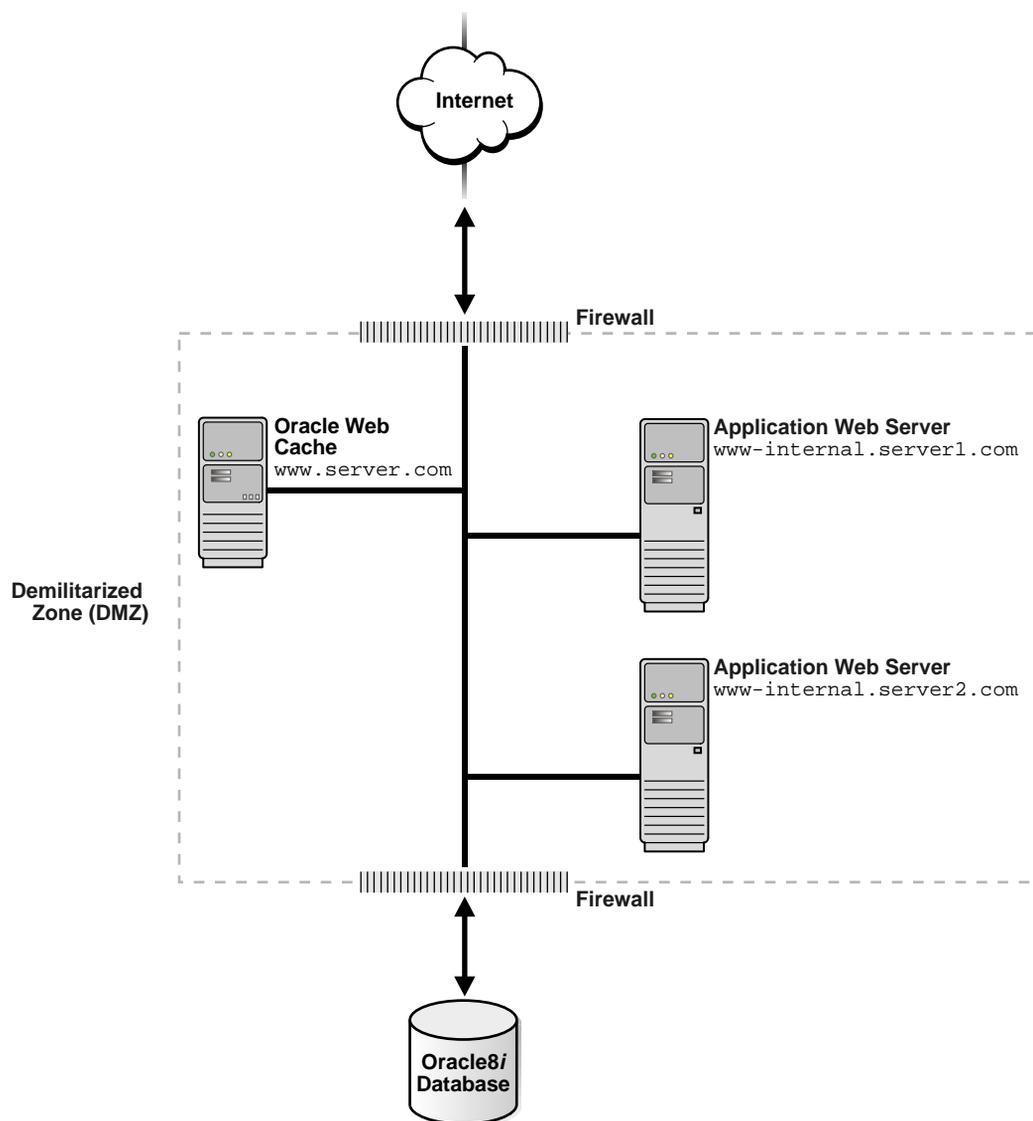
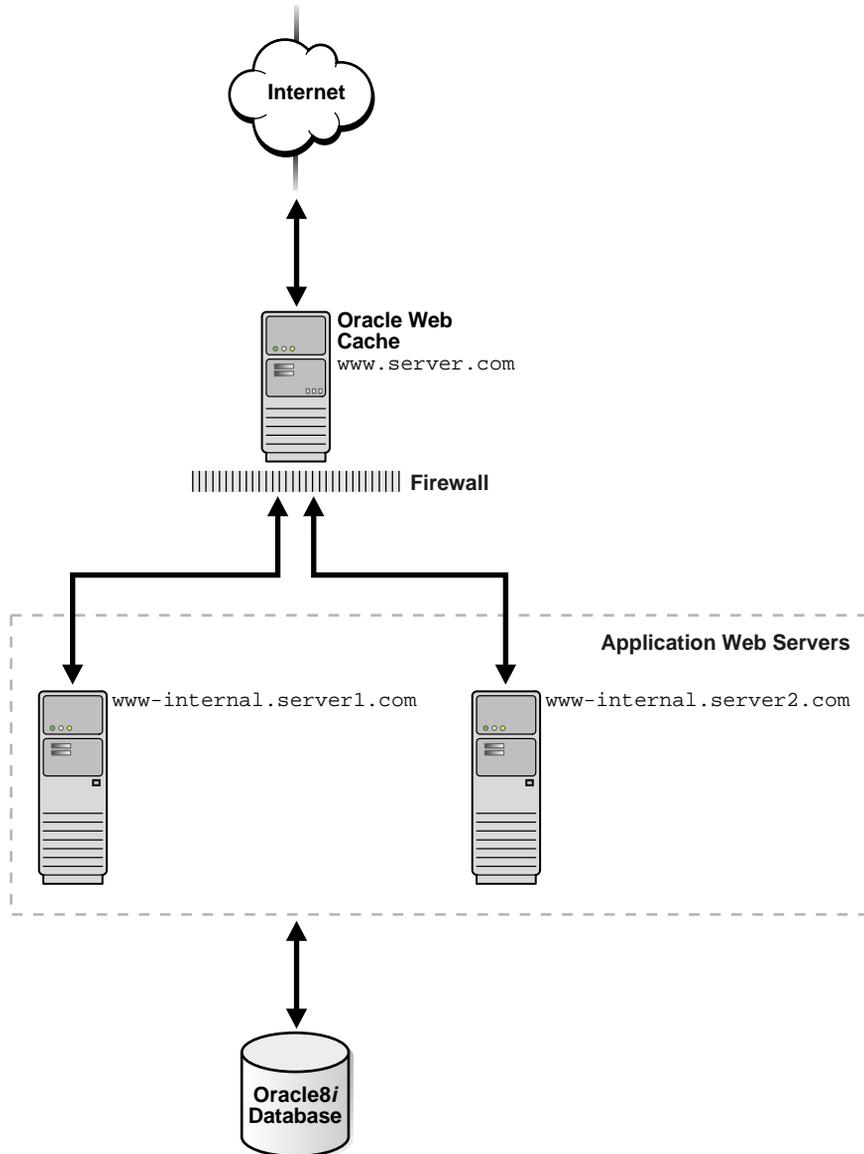
Figure 3–9 *Configuring Oracle Web Cache Inside a Firewall*

Figure 3–10 on page 3-16 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 3-10 *Configuring Oracle Web Cache Outside a Firewall*

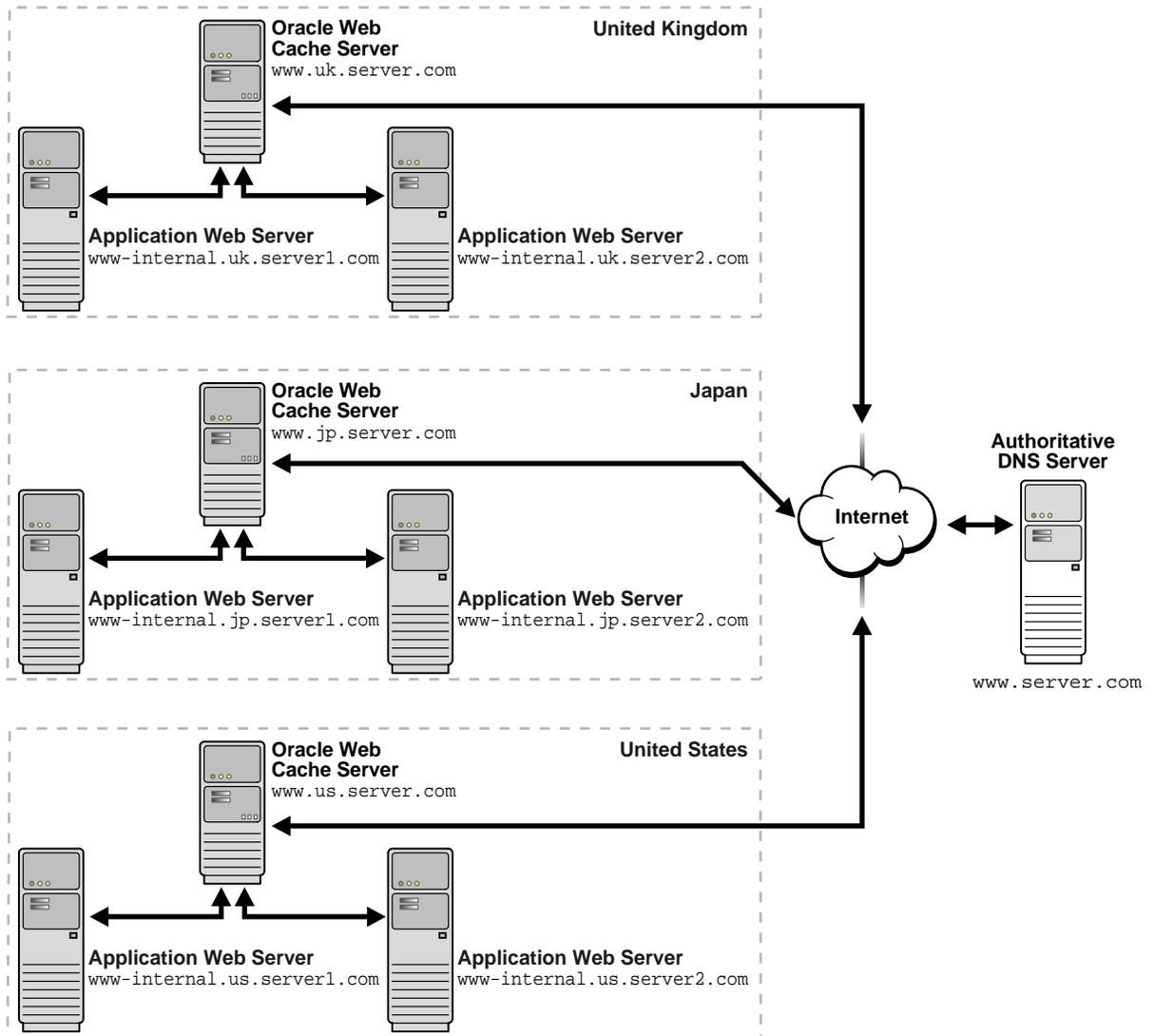


Deploying Oracle Web Cache Servers in a Distributed Network

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 3-11](#) on page 3-18 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 3-11 Distributed Caching

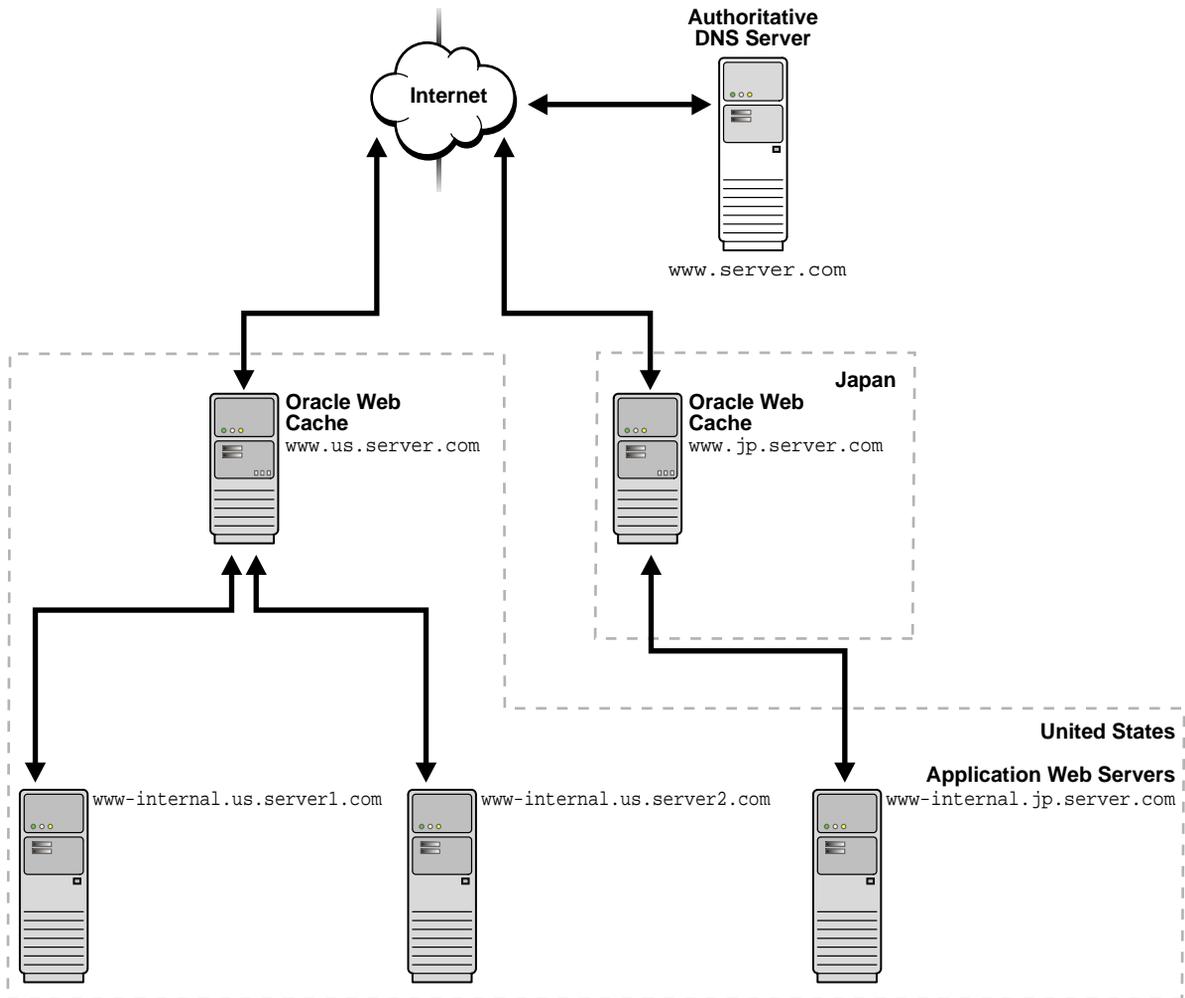


To configure this deployment:

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 3-12](#) on page 3-20. 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 3–12 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 [Figure 3-12](#):

- 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`.

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 username, 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.

See Also:

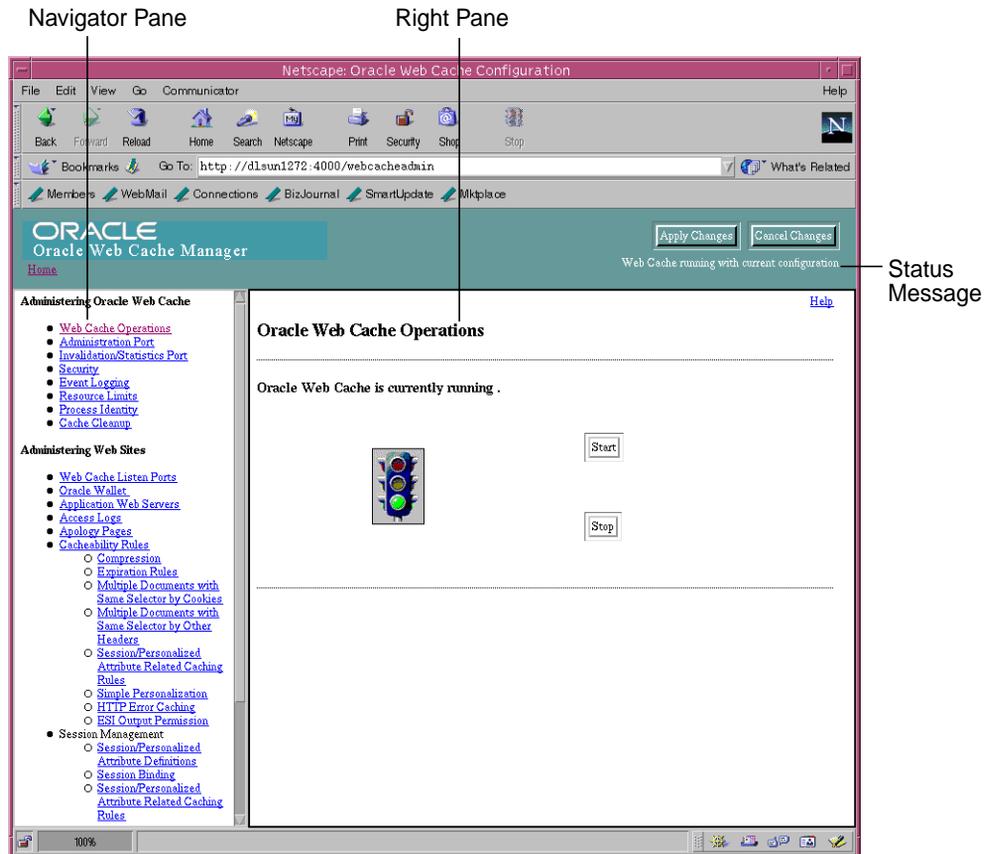
- ["Task 5: \(Optional\) Modify Ports for Administration, Invalidation, and Statistics Monitoring Requests"](#) on page 5-15 for information on modifying port 4000
- ["Task 2: Modify Security Settings"](#) on page 5-2 for information on modifying the administrator's password

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 Manager 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

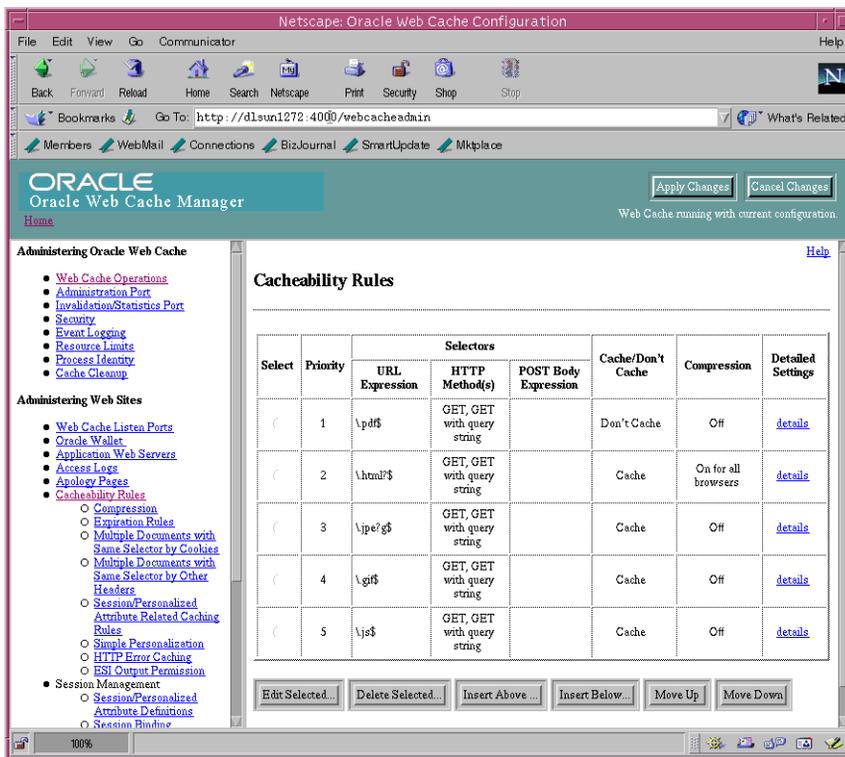
The navigator pane provides a graphical tree view of configuration, administration, and performance monitoring capabilities for Oracle Web Cache and its supported Web sites. The navigator pane contains the following major folders:

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
Configure support for a Web site.	"Task 3: Specify Web Site Settings" on page 5-5
Set the maximum cache size limit.	"Task 4: Set Resource Limits" on page 5-8
Modify listening ports for administration, invalidation, and statistics monitoring requests.	"Task 5: (Optional) Modify Ports for Administration, Invalidation, and Statistics Monitoring Requests" on page 5-15
Configure cacheability rules.	"Configuring Cacheability Rules" on page 6-6
Load balance requests over multiple application Web servers.	"Configuring Load Balancing and Failover" 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 8-25
Configure access log settings.	"Configuring Access Logs" on page 8-31

Task	See Also
Configuring Support for HTTPS Requests	
Configure Oracle Web Cache with an HTTPS listening port.	Step 2 of " Task 3: Specify Web Site Settings " on page 5-5
(Optional) Configure HTTPS listening ports for administration, invalidation, and statistics monitoring requests.	"Task 5: (Optional) Modify Ports for Administration, Invalidation, and Statistics Monitoring Requests" on page 5-15
Create a wallet.	"Task 6: (Optional) Configure the Oracle Wallet" on page 5-17
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.	"Invalidating Documents in the Cache" on page 8-3
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 9-7

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

Part II

Configuration and Administration of Oracle Web Cache

Part II describes how to set up and configure Oracle Web Cache.

This part contains these chapters:

- [Chapter 5, "Initial Setup and Configuration"](#)
- [Chapter 6, "Creating Rules for Cached Content"](#)
- [Chapter 7, "Configuration Considerations for Web Sites with Multiple Application Web Servers"](#)
- [Chapter 8, "Administering Oracle Web Cache"](#)
- [Chapter 9, "Monitoring Performance"](#)
- [Chapter 10, "Troubleshooting Oracle Web Cache Configuration"](#)
- [Chapter 11, "A Case Study Deployment"](#)

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: Specify Web Site Settings](#)
- [Task 4: Set Resource Limits](#)
- [Task 5: \(Optional\) Modify Ports for Administration, Invalidation, and Statistics Monitoring Requests](#)
- [Task 6: \(Optional\) Configure the Oracle Wallet](#)
- [Task 7: Specify Caching Rules](#)
- [Task 8: 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 installed, it is set up with default passwords for administration and invalidation requests. 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. You can enter IP addresses in one of the following formats:

- Complete IP address in dot notation, including the network number, subnet address, and unique host number

Example: 10.1.2.3

- Network/netmask pair for subnet restriction through masking

Example: `10.1.0.0/255.255.0.0` allows all the hosts in the `10.1` subnet access.

- Network/*nnn* Classless Inter-Domain Routing (CIDR) specification to require *nnn* bits from high end to match

Example: `10.1.0.0/16` allows all the hosts in the `10.1` subnet access. This example is similar to the network/netmask example, except the netmask consists of *nnn* high-order 1 bits.

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 can user can execute `webcachel1 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**.

Note: If you changed the password for the administrator user in Step 2, you must restart the **admin server process** with the `webcachel1 start` command when performing "[Task 8: Restart Oracle Web Cache](#)" on page 5-21.

Task 3: 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. Configure the application Web servers for the Web site.

By default, the listening port and host name of the Oracle HTTP Server are configured. Oracle HTTP Server has a default listening port of 7777 on UNIX and 80 on Windows.

- 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.

In multiple application Web server configurations, the capacity of an individual server is weighted against the total capacity of all configured application Web servers. When capacity is set, Oracle Web Cache 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:

$$\text{Application Web Server Capacity} / \text{Total Capacity of All Application Web Servers}$$

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.

$$50/90 = 55\%$$

$$40/90 = 45\%$$

If this is the only application Web server, then 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 considers 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**.

- 2.** Configure a listening port from which Oracle Web Cache will receive browser requests.

By default, Oracle Web Cache listens with the HTTP protocol on port 1100. 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. In addition, you can replace the HTTP listening port to an HTTPS listening port to cache pages for HTTPS requests.

- 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 that this port number is not already in use.

- e.** From the **Protocol** list, select either **HTTP** to accept HTTP browser requests on the port or **HTTPS** to accept HTTPS browser requests on the port.

Note: In this release, Oracle Web Cache has no capability for permitting only HTTPS on an HTTPS protocol. In other words, both HTTP and HTTPS requests will be accepted. If you want only HTTPS requests to be accepted, Oracle Corporation recommends removing HTTP listening ports.

See Also: ["Secure Sockets Layer \(SSL\) Support"](#) on page 1-16

- f. Choose **Submit**.
3. In the Oracle Web Cache Manager main window, choose **Apply Changes**.

Task 4: Set Resource Limits

To set resource limits for Oracle Web Cache, configure the following attributes:

- [Cache Memory](#)
- [Connection Limit](#)

Cache Memory

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. Generally, the amount of memory (maximum cache size) for Oracle Web Cache should be set to at least 256 MB. By default, the maximum cache size is set to 500 MB, which is sufficient for most caches.

To be more precise in determining the maximum amount of memory required, you can take the following steps:

1. Determine what documents you want to cache and the size and type of each document.

One way to do this is to look at existing web server logs for one day to see what documents are popular. From the list of URLs in the log, decide which ones you want to cache. Retrieve the documents and get the size of each document.

2. Calculate the amount of memory needed. The way you calculate it may differ depending on the version of Oracle Web Cache.

The amount of memory that Oracle Web Cache uses to store a document depends on whether the document is larger or smaller than 4 KB:

- If a document is smaller than 4 kilobytes (KB), Oracle Web Cache uses a buffer of 4 KB to store the HTTP body.
- If a document is 4 KB or larger, Oracle Web Cache uses buffers of 32 KB to store the HTTP body. For example, if a document is 40 KB, Oracle Web Cache uses two 32 KB buffers to store the HTTP body.

- Regardless of the size of the body, Oracle Web Cache uses 8 KB to store the HTTP response header.

Use the following formula to determine an estimate of the maximum memory needed:

$$(X * (4KB + 8KB)) + (Y * (([m/32] * 32KB) + 8KB)) + 100MB$$

In the formula:

- X is the number of documents smaller than 4 KB.
- 4KB is size of the buffer for the HTTP body for documents smaller than 4 KB.
- 8KB is the size of the buffer for the HTTP response header.
- Y is number of documents that are 4 KB or larger.
- $[m/32]$ is the ceiling of m (the average size, in kilobytes, of documents 4 KB or larger) divided by 32. A ceiling is the closest integer that is greater than or equal to the number.
- 32KB is size of the buffer for the HTTP body for documents that are 4 KB or larger.
- 8KB is the size of the buffer for the HTTP response header.
- 100MB is the base amount, in megabytes, of memory needed to run Oracle Web Cache. This amount includes memory for internal functions such as lookup keys and timestamps.

For example, assume that you want to cache 5000 documents that are smaller than 4 KB and 2000 documents that are 4 KB or larger and that the larger documents have an average size of 54 KB. Use the formula to compute the maximum memory:

$$(5000 * (4KB + 8KB)) + (2000 * (([54/32] * 32KB) + 8KB)) + 100MB$$

Using the formula, you need:

- 60,000 KB for the smaller documents.
- 144,000 KB for the larger documents. For the HTTP body, you need 64 KB (two 32 KB buffers) for each document, given the average size of 54 KB. For the HTTP response header, you need 8 KB for each document.
- 100 MB for the base amount of memory needed for Oracle Web Cache.

This results in an estimate of 300 MB of memory needed.

Note that this formula does not take into account the complexities of calculating the size of **Edge Side Includes (ESI)** pages. The only valid method to calculate the size of pages cached with ESI is to monitor the cache.

3. Configure Oracle Web Cache, specifying the result of the formula as the maximum cache size. Remember that the result is only an estimate.

To specify 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 result of the formula.

- d. Choose **Submit**.

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

Even though you have specified that certain documents should be cached, not all of the documents are cached at the same time. Only those documents that have been requested and are valid are stored in the cache. As a result, only a certain percentage of your documents are stored in the cache at any given time. That means that you may not need the maximum memory derived by the preceding formula.

4. Start Oracle Web Cache and, using a simulated load or an actual load, monitor the cache to see how much memory it really uses in practice.

Remember that the cache is empty when Oracle Web Cache starts. For monitoring to be valid, make sure that the cache is fully populated. That is, make sure that the cache has received enough requests so that a representative number of documents are cached.

The Oracle Web Cache Statistics and the Oracle Web Cache Detailed Statistics Monitor pages provide information about the current memory use and the maximum memory use.

- a. To access the Oracle Web Cache Statistics page, from the navigator pane, select **Administering Web Sites > Monitoring Oracle Web Cache > Statistics**.

The **Cache Size (in bytes)** metric shows the current logical size of the cache. The logical size of the cache is the size of the valid documents in the cache. For example, if the cache contains two documents, one 3 KB and one 50 KB, the cache size is 53 KB, the total of the two sizes. This metric does not show the physical size of the cache.

- b. To access the Oracle Web Cache Detailed Statistics Monitor page, enter the following URL:

`http://web_cache_hostname:statistics_port`

By default, the statistics port is set to port 4002.

The Oracle Web Cache Detailed Statistics Monitor page displays the physical size of the cache. The physical size of the cache is the amount of memory used by Oracle Web Cache.

In the Oracle Web Cache Memory Manager Block Usage table, observe the **Current Action Limit** and the **Current Allocated Memory** metrics. The **Current Allocated Memory** metric is usually smaller than the **Current Action Limit** metric. If the **Current Allocated Memory** is close to or greater than the **Current Action Limit**, increase the maximum cache size.

Connection Limit

In addition to the cache size, it is also important to specify the maximum connection limit to the Oracle Web Cache server. When you configure this limit, set a reasonable number. If you set a number that is too high, then performance can be affected. To help determine the correct number, use various tools available with the operating system. For example, the `netstat -a` command on UNIX and Windows enables you to determine the number of established connections.

Note that you should set the maximum number of incoming connections in Oracle Web Cache Manager to correctly reflect the maximum number of clients you intend to serve concurrently at any given time. Do not set the value to an arbitrary high value, because Oracle Web Cache sets aside some resources for each connection, adversely affecting performance.

To set the maximum number incoming connections limit:

1. In the navigator pane, select **Administering Oracle Web Cache > Resource Limits**.

The Resource Limits page appears in the right pane.

2. In the Resource Limits page, choose **Change connections limit**.

The Change Maximum Incoming Connections Limit dialog box appears.

3. In the **New maximum connections limit** field, enter the new limit.
4. Choose **Submit**.
5. In the Oracle Web Cache Manager main window, choose **Apply Changes**.

Connections on UNIX

On most UNIX operating systems, each client connection requires a separate file descriptor. The number of file descriptors that a process can open is usually set to 1024. You can change this number to a higher one, but you must change the ownership of the executable `webcached` to root and make it executable to all users. In addition, you must add set-user ID permission to the executable. If the ownership of `webcached` is not changed, Oracle Web Cache fails to start and writes the events to the `event_log` file.

Oracle Web Cache uses the following formula to calculate the maximum number of file descriptors to be used:

$$\text{Max_File_Desc} = \text{Current_Max_Conn} + \text{App_Web_Server_Capacity_Sum} + 20$$

In the formula:

- `Max_File_Desc` is the maximum number of file descriptors to be used.
- `Current_Max_Conn` is the current maximum incoming connections limit for Oracle Web Cache. Set the maximum number of incoming connections using the **Administering Oracle Web Cache > Resource Limits** page of Oracle Web Cache Manager.
- `App_Web_Server_Capacity_Sum` is the sum of the capacity for all configured application Web Servers. Set the capacity using the **Administering Web Sites > Application Web Servers** page of Oracle Web Cache Manager.
- 20 is the number of connections reserved for internal use by Oracle Web Cache.

The Oracle Web Cache server tries to reserve the maximum number of file descriptors (`Max_File_Desc`) when it starts. If it fails to do so, it does not default to a lower value but logs an error message and fails to start. If the Oracle Web Cache server fails to start because the number of file descriptors required is more than 1024, you must change the ownership of the executable `webcached` to root and make it executable to all users. In addition, you must add set-user ID permission to the executable.

If the ownership of `webcached` is not changed, then Oracle Web Cache fails to start and writes the following events to the `event_log` file:

```
20/May/2001:18:18:24 -0800 -- Error: Could not increase number of file/socket
descriptors to 10220.
20/May/2001:18:18:24 -0800 -- Error: Failed to start the server.
```

See Also:

- ["Greater Than One Thousand Maximum Connections"](#) on page 10-7 for information on changing the ownership of `webcached` to root
- Operating system-specific documentation for connection limitations
- Operating system-specific *Oracle HTTP Server powered by Apache Performance Guide* for TCP/IP performance tuning tips

Connections on Windows

On Windows NT and 2000, the number of file handles as well as socket handles is limited only by available kernel resources, more precisely, by the size of paged and non-paged pools. However, the number of active TCP/IP connections is restricted by the number of TCP ports the system can open.

The default maximum number of TCP ports is set to 5000 by the operating system. Of those, 1024 are reserved by the kernel. You can modify the maximum number of ports by editing the Windows registry. Windows NT and Windows 2000 allow up to 65536 ports.

To change the default, you must add a new value to the following registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
```

Add a new value, specifying the following:

- Value Name: MaxUserPort
- Data Type: REG_DWORD
- Data: An integer less than 65536 - 1024

Do not set the maximum number of incoming connections in Oracle Web Cache Manager to a number greater than the number of TCP ports minus 1024. On Windows, Oracle Web Cache does not attempt to reserve file handles or to check that the number of current maximum incoming connections is less than the number of TCP ports.

See Also:

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

Task 5: (Optional) Modify Ports for Administration, Invalidation, and Statistics Monitoring Requests

In addition to receiving HTTP and HTTPS browser request, Oracle Web Cache also receives administration, invalidation, and statistics monitoring requests on specific HTTP or HTTPS listening ports:

```
http://web_cache_hostname:http_port  
https://web_cache_hostname:https_port
```

By default, Oracle Web Cache uses the HTTP protocol to receive these requests. Default port numbers are as follows:

- 4000 for administration and configuration requests from Oracle Web Cache Manager
- 4001 for invalidation requests
- 4002 for statistics monitoring requests

Changing the Default Administration Listening Endpoint

To change the default port number or protocol for administration requests or configuration changes:

1. In the navigator pane, select **Administering Oracle Web Cache > Oracle Web Cache Administration Port**.

The Oracle Web Cache Administration Port page appears in the right pane.

2. In the Oracle Web Cache Administration Port page, choose **Edit**.

The Change Administration Port dialog box appears.

3. In the **New Administration Port** field, enter the new port.
4. From the **Protocol** list, select either **HTTP** or **HTTPS** to accept administration requests from one of the following URLs:

```
http://web_cache_hostname:administration_port  
https://web_cache_hostname:administration_port
```

5. Choose **Submit**.
6. In the Oracle Web Cache Manager main window, choose **Apply Changes**.

Note: When changing the administration port, note the following:

- Requests to the administration port must originate from a trusted host or a host on a trusted subnet. Trusted hosts and subnets are defined in the Security page (**Administering Oracle Web Cache > Security**). See "[Task 2: Modify Security Settings](#)" on page 5-2 for further information.
 - Restart the `admin` server process with the `webcachectl start` command when performing "[Task 8: Restart Oracle Web Cache](#)" on page 5-21.
-
-

Changing the Default Invalidation and Statistics Monitoring Listening Endpoints

To change the default port number or protocol for invalidation or statistics monitoring requests:

1. In the navigator pane, select **Administering Oracle Web Cache > Oracle Web Cache Administration Port**.

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

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

The Change Invalidation/Statistics Port dialog box appears.

3. In the **New Invalidation Port** field, enter the new port.
4. From the **Protocol** list, select either **HTTP** or **HTTPS** to accept invalidation requests from one of the following URLs:

```
http://web_cache_hostname:invalidation_port  
https://web_cache_hostname:invalidation_port
```

5. From the **Protocol** list, select either **HTTP** or **HTTPS** to accept statistics monitoring requests from one of the following URLs:

```
http://web_cache_hostname:statistics_port  
https://web_cache_hostname:statistics_port
```

Note: If you enable HTTPS for invalidation or statistics monitoring requests, you cannot use the Oracle Web Cache Manager interface to send the requests. You must submit requests through the URL.

6. Choose **Submit**.
7. In the Oracle Web Cache Manager main window, choose **Apply Changes**.

Task 6: (Optional) Configure the Oracle Wallet

If you configured Oracle Web Cache to receive HTTPS browser requests in Step 2 of "[Task 3: Specify Web Site Settings](#)" on page 5-5, then create a wallet for server-side certification on the computer running Oracle Web Cache. The wallet manages authentication data such as keys, certificates, and trusted certificates needed by the [Secure Sockets Layer \(SSL\)](#). Once the wallet is created, use Oracle Web Cache Manager to specify the location of the wallet.

This section covers the following topics relating to wallet configuration:

- [Creating a Wallet](#)
- [Specifying the Location of Wallets](#)
- [Enabling Wallets to Open on Windows](#)

Creating a Wallet

To create the wallet, use Oracle Wallet Manager.

To start Oracle Wallet Manager:

- On UNIX, run `owm` from `$ORACLE_HOME/bin`
- On Windows, choose **Start > Programs > Oracle - *HOME_NAME* > Network Administration > Wallet Manager**

See Also:

- ["Secure Sockets Layer \(SSL\) Support"](#) on page 1-16
- *Oracle9i Application Server Oracle Wallet Manager User's Guide*

Specifying the Location of Wallets

Once the wallet is created, use Oracle Web Cache Manager to specify the location of the wallet. You specify the wallet for the Oracle Web Cache computer when you configure listening ports. Listening ports for a site can share the same wallet.

Wallets have the following default locations:

- `/etc/ORACLE/WALLETS/user_name` on UNIX
- `%USERPROFILE%\ORACLE\WALLETS` on Windows NT and Windows 2000

To change the default location of the wallet:

1. In the navigator pane, select **Administering Web Sites > Oracle Wallet**.
The Oracle Wallet page appears in the right pane.
2. In the Oracle Wallet page, choose **Edit**.
The Change Oracle Wallet dialog box appears.
3. In the **Wallet Resource Locator** field, enter the location of the wallet.
On UNIX, use `file:` to prefix the directory path. On Windows, use `file:drive_letter:` to prefix the directory path.
4. Choose **Submit**.
5. In the Oracle Web Cache Manager main window, choose **Apply Changes**.

Enabling Wallets to Open on Windows

Oracle Web Cache attempts to open wallets at startup on Windows NT and 2000. On Windows NT and 2000, wallets are protected so that only the user that created them can open and use them. By default, Oracle Web Cache services are associated with the local system account, which does not have permission to open wallets.

To enable Oracle Web Cache to open wallets at startup:

1. Create a wallet with an administrator account.
2. Change the system account information for the Oracle Web Cache services:

Windows NT	Windows 2000
<ol style="list-style-type: none"> 1. Choose the Services icon from the Control Panel window. The Services window appears. 2. Select the OracleHOME_NAMEWebCache service. The Service dialog appears. 3. Choose This Account. By default the LocalSystem user account is associated with the service. 4. Choose "..." next to This Account. The Add User dialog box appears. 5. Select the user that created the wallet from the Names list, and then choose Add. 6. Choose OK to close the Add User dialog box. 7. In the Service dialog box, provide the password for the wallet administrator in the Password field, and then confirm the password in the Confirm Password field. 8. In the Services dialog box, choose OK. 9. Repeat Steps 3 - 9 for the OracleHOME_NAMEWebCacheAdmin and OracleHOME_NAMEWebCacheMon services. 10. In the Services window, choose Close. 	<ol style="list-style-type: none"> 1. Choose Administrative Tools > Services from the Control Panel window. The Services window appears. 2. Select the OracleHOME_NAMEWebCache service. The OracleHOME_NAMEWebCache Properties dialog appears. 3. Choose the Log On tab. 4. In the Log On tab, choose This account. By default the LocalSystem user account is associated with the service. 5. Choose Browse next to This Account. The Select User dialog box appears. 6. Select the user that created the wallet from the list, and then choose OK. 7. Choose OK to close the Add User dialog box. 8. In the OracleHOME_NAMEWebCache Properties dialog box, provide the password for the wallet administrator in the Password field, and then confirm the password in the Confirm Password field. 9. In the Services dialog box, choose OK. 10. Repeat Steps 3 - 9 for the OracleHOME_NAMEWebCacheAdmin and OracleHOME_NAMEWebCacheMon services.

3. On Windows NT, grant the wallet administrator the right to run Oracle Web Cache as a service.
 - a. Choose **Start > Programs > Administrative Tools > User Manager**.
The User Manager window appears:
 - b. Select the wallet administration, and then choose **Policies > User Rights**.
The User Rights Policy dialog box appears:
 - c. Choose the **Show Advanced User Rights** check box, and then select **Log on as a service** from the **Right** list.
 - d. Select **Users** from the **Grant To** list.
If **Users** does not exist, create it:
 1. Choose **Add**.
The Add Users and Groups dialog box appears:
 2. Select the name of the local host computer from the **List Names From** list.
 3. Select **Users** from the **Names** list, and then choose **Add**.
 4. Choose **OK**.
Users appears in the **Grant To** list.
 - e. Choose **OK** in the User Rights Policy dialog box.
The User Manager window reappears.
 - f. Choose **User > Exit**.

See Also: ["Wallet Cannot Be Opened"](#) on page 10-8

Task 7: Specify Caching Rules

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

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

Task 8: Restart Oracle Web Cache

When Oracle Web Cache is configured, stop it and start it again to read in the new configuration settings. When you stop Oracle Web Cache, all objects are cleared from the cache. In addition, all statistics are cleared. 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 4-2 2. 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>

Note: If you change the administration password in the Security page (**Administering Oracle Web Cache > Security**) or the administration listening port in the Administration port page (**Administering Oracle Web Cache > Administration Port**), you must restart the `admin` server process with the `webcachectl start` command.

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

Creating Rules for Cached Content

This chapter explains how to configure cacheability rules.

This chapter contains these topics:

- [Cacheability Rules Overview](#)
- [Configuring Cacheability Rules](#)
- [Configuring Expiration Rules](#)
- [Configuring Rules for Multiple-Version Documents Containing Cookies](#)
- [Configuring Rules for Multiple-Version Documents Containing HTTP Request Headers](#)
- [Configuring Rules for Pages with Simple Personalization](#)
- [Configuring Rules for Pages with Session Tracking](#)
- [Configuring Pages for Content Assembly and Partial Page Caching](#)
- [Configuring Cacheability Attributes in Response Headers](#)

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 documents, personalized pages, pages that support session tracking, and HTTP error messages.

This section discusses the following topics:

- [Cacheability Rule Creation](#)
- [Cacheability Rule Syntax](#)
- [Default Cacheability Rules](#)

Cacheability Rule Creation

Generally, when you assign cacheability rules, you specify the **regular expression** matching the URL and whether you want the documents contained within the URL cached or not cached. You then order the cacheability rules in order of priority. Higher priority rules are matched 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`, GET and GET with query string, Don't Cache
2. `^/cec/cstage\?ecaction=ecpassthru.*`, GET and GET with query string, Cache

Note that GET and GET with query string are the **HTTP request methods** used by the documents.

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 updating transactions, shopping cart views, personal account views, and so on. 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.

In addition to the URL, you can specify optional **selectors** for more fine-grained cacheability rules. These additional selectors include the HTTP request method (GET, GET with query string, or POST) and, if POST is selected, the HTTP POST body of the documents. In the following rule list, Rule 2 caches documents of the URL that use the GET and GET with query string methods, and Rule 3 caches documents of the URL that use the POST method and a POST body matching `action=search`.

1. `^/cec/cstage\?ecaction=ecpassthru2`, GET and GET with query string, Don't Cache
2. `^/cec/cstage\?ecaction=ecpassthru.*`, GET and GET with query string, Cache
3. `^/cec/cstage\?ecaction=ecpassthru.*`, POST, `action=search`, Cache

Note: 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

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 caret (^) 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.gift`.

- Use a period (.) to match any one character
- Use a question mark (?) to match zero or one occurrence of the character that it follows.
- Use an asterisk (*) to match zero or more occurrences of the pattern that it follows.
- 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/regex_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 <code>/machine/doc</code> and ending in <code>*.gif</code>	<code>^/machine/doc/*.gif\$</code>
All Graphics Interchange Format (GIF) images	<code>\.gif\$</code>
<code>/robots.txt</code> file	<code>^/robots.txt\$</code>
All procedures in the <code>new_employee</code> package	<code>^/pls/enroll_db/new_employee</code>

Default Cacheability Rules

Figure 6–1 displays the default cacheability rules established when Oracle Web Cache is installed.

Figure 6–1 Default Cacheability Rules

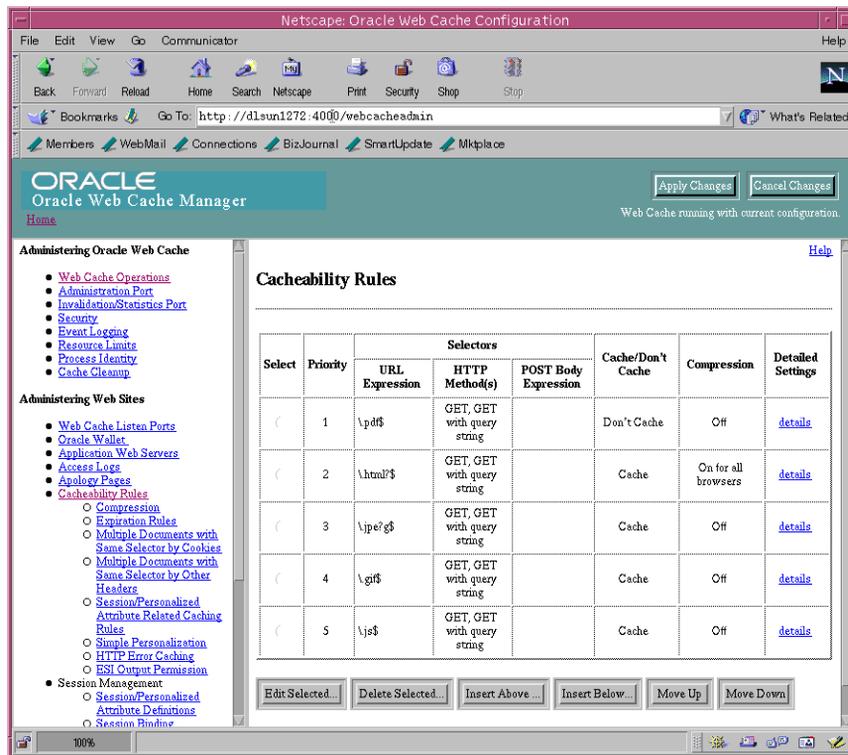


Table 6–2 describes the default cacheability rules.

Table 6–2 Default Cacheability Rules

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

URL Expression	Cache/Don't Cache	Description
\.jpe?g\$	Cache	Instructs Oracle Web Cache to cache .jpg and .jpeg (JPEG) files
\.gif\$	Cache	Instructs Oracle Web Cache to cache .gif (GIF) files
\.js\$	Cache	Instructs Oracle Web Cache to cache .js (JavaScript) files

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. In the **Method** section, select to cache documents that use `GET`, `GET` with query string, or `POST` HTTP request methods.

You select more than one request method.

Note: 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.

6. If you selected **POST** in Step 5, specify the HTTP `POST` body of the documents in the **POST Body Expression** field.
To apply this rule to any `POST` request body, enter ".*" in the field.
7. Select **Cache** or **Don't Cache** for the documents contained within the URL.
8. Optionally, to help track the meaning of rules, enter a comment for the cacheability rule in the **Comment** field.
9. In **ESI Output Permission**, select either **Yes** or **No**.
 - Select **Yes** to enable **Edge Side Includes (ESI)**-compliant proxy caches, such as Akamai EdgeSuite, to process ESI tags. Select **Yes** only if the following conditions apply:
 - The ESI-compliant cache or service resides between browsers and Oracle Web Cache
 - You prefer the cache or service to perform the ESI processing rather than Oracle Web Cache.
 - Select **No** to disallow other ESI-compliant caches or services from processing ESI tags.

See Also: ["Configuring Pages for Content Assembly and Partial Page Caching"](#) on page 6-33

10. Select options for the columns that apply, and then choose **Submit**:

Compression

Select **No** to not serve compressed cacheable and non-cacheable documents for browsers.

Select **Yes** to serve compressed cacheable and non-cacheable documents for browsers, and then select one of the following options:

- **Compress for all browsers** to serve compressed documents to all browser types
- **Compress for non-Netscape browsers only** to serve compressed documents for all browsers other than Netscape

Important: Netscape browsers are unable to uncompress included files, which may result in Netscape failures. If a document will be included in other files, such as a JavaScript file, then select **Compress for non-Netscape browsers only**.

Notes:

- Oracle Corporation recommends not compressing images, such as GIFs and JPEGs, as well as executables and files that are already zipped with utilities like WinZip and GZIP. Compressing these files incurs additional overhead without the benefits of compression.
- Even if compression is turned on, Oracle Web Cache does not compress documents containing the following:
 - A `Content-Encoding` header, which is typically used to denote compression
 - Session-encoded URLs, the `<!--WEBCACHETAG-->` and `<!--WEBCACHEEND-->` tags, ESI tags, or other dynamic content that requires modification after compression

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, then choose **Create A New Rule** to create a new rule.

See Also: Step 4 in "[Configuring Expiration Rules](#)" on page 6-14

Multiple Documents with the Same Selector by Cookies Select **None** to not have Oracle Web Cache cache multiple-version documents that use cookies.

Select **Apply the following** to cache multiple-version documents that rely on **category cookie** values, and then select the required cookie rules. If you do not see a cookie rule that can be applied to these documents, then 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 Documents Containing Cookies](#)" on page 6-16

Multiple Documents with the Same Selector by Other Headers Select the **HTTP request headers** whose values Oracle Web Cache will use to cache and identify multiple-version documents.

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-xbitmap, 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 an HTTP request header of `User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0)` and another request sends an 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/Personalized Attribute 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 cache documents with session information, and then select the required session rules. If you do not see the session these documents require, then choose **Create A New Rule** to create a new rule.

See Also: "[Configuring Rules for Pages with Session Tracking](#)" on page 6-28 for further information about creating session-related caching rules

Simple Personalization 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:

- **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.

See Also: "[Configuring Rules for Pages with Simple Personalization](#)" on page 6-19 for further information about creating rules for personalized pages

HTTP Error Caching Enter the HTTP error 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, then 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 8-3

11. Repeat Steps 3 through 10 for each cacheability rule.

Once the cacheability rules are configured, prioritize them.

To assign priority to rules:

1. In the Cacheability Rules page, select a cacheability rule, and then choose **Move Up** or **Move Down** to order the rules.

Higher priority rules are processed first.

2. 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 6–2 illustrates how an administrator might set up cacheability rules for a catalog built on Oracle iStore 3i technology, which enables e-merchants to design, build, and publish their stores on the World Wide Web. Note that rules for Oracle iStore 11i will differ.

Figure 6–2 Cacheability Rules Example

Cacheability Rules

Select	Priority	Selectors			Cache/Don't Cache	Compression	Detailed Settings
		URL Expression	HTTP Method(s)	POST Body Expression			
<input type="checkbox"/>	1	template-walkin\en\htm	GET, GET with query string		Cache	On for all browsers	details
<input type="checkbox"/>	2	epassthru2	GET, GET with query string		Don't Cache	On for all browsers	details
<input type="checkbox"/>	3	epassthru	GET, GET with query string		Cache	On for all browsers	details
<input type="checkbox"/>	4	\.jpg\$	GET, GET with query string		Cache	Off	details
<input type="checkbox"/>	5	\.gif\$	GET, GET with query string		Cache	Off	details
<input type="checkbox"/>	6	\.html\$	GET, GET with query string		Cache	On for all browsers	details
<input type="checkbox"/>	7	\.js\$	GET, GET with query string		Cache	Off	details
<input type="checkbox"/>	8	ecprdrumsearch	GET, GET with query string		Cache	On for all browsers	details
<input type="checkbox"/>	9	eclogout	GET, GET with query string		Don't Cache	On for all browsers	details
<input type="checkbox"/>	10	eccartview	GET, GET with query string		Don't Cache	On for all browsers	details
<input type="checkbox"/>	11	ecacctroot	GET, GET with query string		Don't Cache	On for all browsers	details
<input type="checkbox"/>	12	\.pdf\$	GET, GET with query string		Don't Cache	Off	details
<input type="checkbox"/>	13	*	GET, GET with query string		Cache	On for all browsers	details

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

Table 6-3 Regular Expression Examples

Priority Order	URL Expression	HTTP Method(s)	Cache/Don't Cache	Description
1	template=walkin\.en\.htm	GET, GET with query string	Cache	Caches the home page, including personalized information
2	ecpassthru2	GET, GET with query string	Don't Cache	Does not cache this template, because it contains customized user account information
3	ecpassthru	GET, GET with query string	Cache	Caches this template, because it does not contain customized content
4	\.jpe?g\$	GET, GET with query string	Cache	Caches .jpg and .jpeg files
5	\.gif\$	GET, GET with query string	Cache	Caches .gif files
6	\.html?\$	GET, GET with query string	Cache	Caches all .htm and .html files
7	\.js\$	GET, GET with query string	Cache	Caches .js files
8	ecprditmsearch	POST	Cache	Caches the search results
9	eclogout	GET, GET with query string	Don't Cache	Does not cache logout results
10	eccartview	GET, GET with query string	Don't Cache	Does not cache the shopping cart
11	ecacctroot	GET, GET with query string	Don't Cache	Does not cache the account view
12	\.pdf\$	GET, GET with query string	Don't Cache	Does not cache pages ending in .pdf
13	.*	GET, GET with query string	Cache	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:

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.

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.

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.
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.

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, select the newly-created rule, and then choose **Change Selector Association**.

The Change Policy-Selector Association dialog box appears.

9. Select a selector from the right list, and then choose the **Make Association** button.

The selector moves to the left list and the dialog box closes.

If the selector you require does not exist, then create a cacheability rule, as described in "[Configuring Cacheability Rules](#)" on page 6-6. In Step 10 of the procedure, select an expiration rule in the **Expiration Rule** row of the Edit/Create Cacheability Rule dialog box.

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 Documents Containing Cookies

See Also: ["Multiple Versions of the Same Document"](#) on page 2-8 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 documents.

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 the Same Selector by Cookies**.

The Multiple Documents with the Same Selector by Cookies page appears in the right pane.

3. In the Multiple Documents with the Same Selector by Cookies page, choose **Create** or **Add**.

The Edit/Create Multiple Documents with the Same Selector by Cookies Rule dialog box appears.

4. In the **Enter cookie name** field, enter the name of the cookie.

5. For the prompt **Also cache documents whose requests do not contain this cookie?**, select either **Yes** or **No**.
 - Select **Yes** to cache versions of the document that do not contains this cookie. This option enables Oracle Web Cache to serve documents from the cache for browser requests that do not use contain this cookie
 - Select **No** to not cache versions of documents that do not contain this cookie.
6. In the Edit/Create Multiple Documents with the Same Selector by Cookies Rule dialog box, choose **Submit**.
7. In the Multiple Documents with the Same Selector by Cookies page, select the newly-created rule, and then choose **Change Selector Association**.

The Change Policy-Selector Association dialog box appears.

8. Select a selector from the right list, and then choose the **Make Association** button.

The selector moves to the left list and the dialog box closes.

If the selector you require does not exist, then create a cacheability rule, as described in "[Configuring Cacheability Rules](#)" on page 6-6. In Step 10 of the procedure, select **Apply the following** and a rule in the **Multiple Documents with the Same Selector by Cookies** row of the Edit/Create Cacheability Rule dialog box.

9. Repeat Steps 3 through 9 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 Documents Containing HTTP Request Headers

See Also: ["Multiple Versions of the Same Document"](#) on page 2-8 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 the Same Selector by Other Headers** column of the Edit/Create Cacheability Rule dialog box.

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

Configuring Rules for Pages with Simple Personalization

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

Personalized attributes are often in the form of a personalized greeting like "Hello, *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 2-12 for an overview and an example scenario
- ["Session-Encoded URLs"](#) on page 2-15 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 Create Session/Personalized Attribute Definition dialog box appears.
 - c. In the **Session/Attribute** 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 3 KB.

- e. In the **Default Value (Optional)** field, enter a default string that Oracle Web Cache will use for the cookie or embedded URL parameter value. Oracle Web Cache uses the default string for those requests without the cookie or parameter information. For these requests, Oracle Web Cache substitutes the personalized attribute or session ID information with the default string. The default string defaults to `default`.

Note: 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 6-28.
 2. In Step 6 of the procedure, select **YES** as the response.
 3. In Step 7 of the procedure, select **NO** as the response.
-
-

- f. In the **Comment** field, enter a description of the definition.
- g. Choose **Submit**.

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

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

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

Important: The `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags cannot be used on a page that contains ESI tags for content assembly and **partial page caching**. If you require simple personalization and are using ESI, see ["Using ESI for Simple Personalization"](#) on page 6-34.

Note: The `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags can appear anywhere the `<!-- ...-->` comment tags are permitted in HTML. For example, you can use the `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags between other HTML tag pairs, but you cannot use them within an HTML tag.

In the following example, the placement of `<!-- WEBCACHETAG="p_name"-->` within the `<input>` tag is an invalid use of the `<!-- WEBCACHETAG-->`:

```
http.p('<form action="test" method="GET">');
http.p('<table border="0" >
    <tr>
        <td><input type="text" name="p_name" size="8"
value="<!--
    WEBCACHETAG="p_name"-->' | |p_name| |'<!--
WEBCACHEEND-->"></td>
        </tr>
    <tr>
        <td><input type="submit" value="Search"></td>
    </tr>
</table>');
```

To achieve personalization within an HTML tag, use ESI.

See Also: ["Example of Simple Personalization with Variable Expressions"](#) on page 6-44

4. Create a cacheability rule for the personalized pages, as described in "[Configuring Cacheability Rules](#)" on page 6-6.

In Step 10 of the procedure, select **Yes** in the **Simple Personalization** row of the Edit/Create Cacheability Rule dialog box, and then select one of the following options:

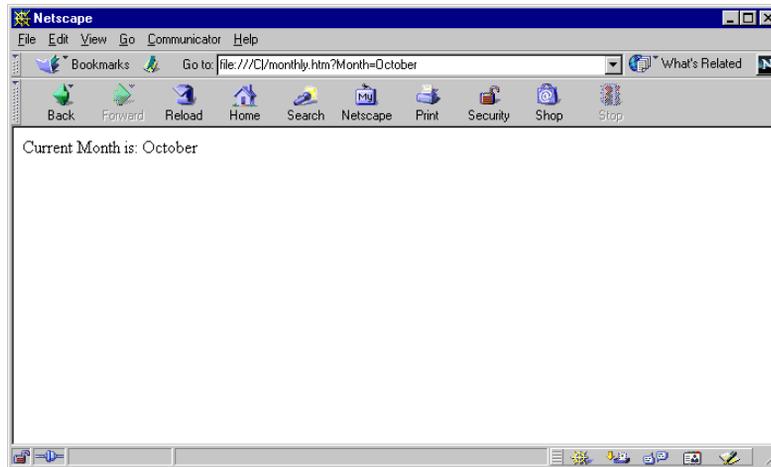
- **pages do not contain HREFs that are session encoded URLs** to cache substitution instructions for only personalized attributes
 - **pages contain HREFs that are session encoded URLs** to cache substitution instructions for both personalized attributes and session-encoded URLs
5. Apply changes and 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 6-3](#).

Figure 6-3 *monthly.htm*



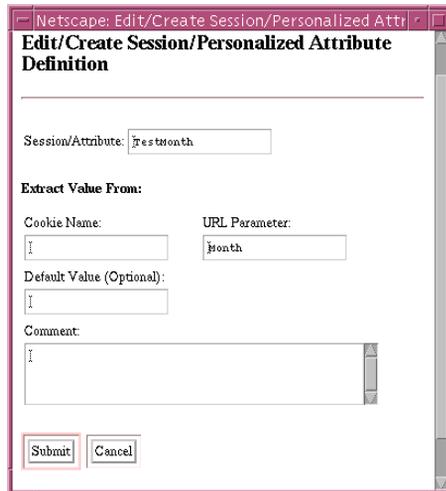
October is personalized content that can be substituted with other values.

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.

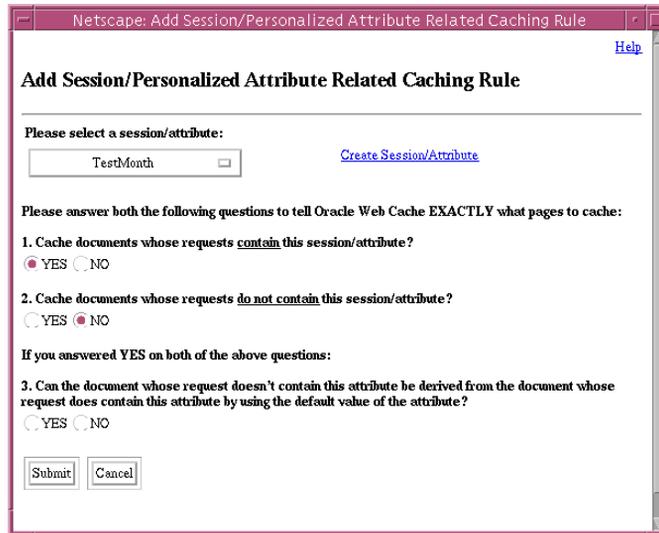
1. A personalized attribute of `TestMonth` was mapped to the embedded URL parameter `Month` in the Edit/Create Session/Personalized Attribute Definition dialog box.

Figure 6–4 *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/Personalized Attribute Related Caching Rule dialog box.

Figure 6-5 Add Session/Personalized Attribute Related Caching Rule Dialog Box



See Also: "[Configuring Rules for Pages with Session Tracking](#)" on page 6-28 for more information about creating personalized attribute caching rules

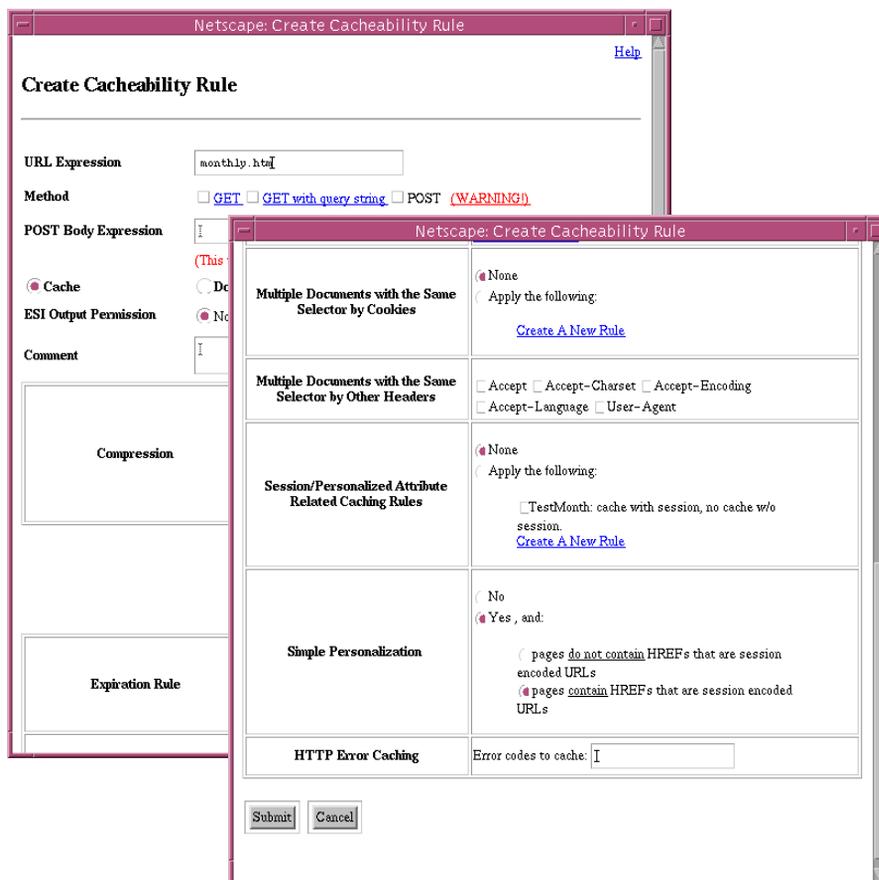
3. The `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` HTML tags were added to `monthly.htm`.

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/Personalized Attribute Related Caching Rules** row, the personalized attribute caching rule for the embedded URL `Month` was chosen.
 - b. In the **Simple Personalization** 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 6–6 Create Cacheability Rule Dialog Box



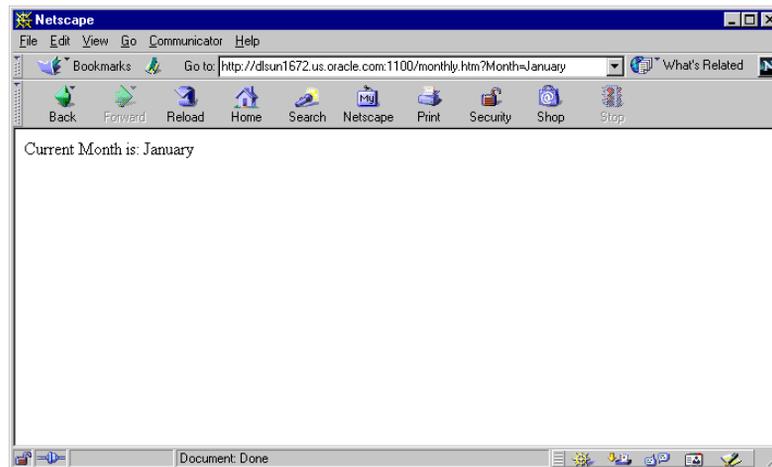
5. The configuration changes are applied:
 - a. In the Oracle Web Cache Manager main window, **Apply Changes** is chosen.
 - b. In the Oracle Web Cache Operations page, Oracle Web Cache is stopped and then restarted.

To verify that Oracle Web Cache was caching `monthly.htm`:

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 6–7 *monthly.htm When Cached*



Configuring Rules for Pages with Session Tracking

See Also: ["Session Tracking"](#) on page 2-14 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: 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** or **Session Management > Session/Personalized Attribute Related Caching Rules**.

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

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

The Add Session/Personalized Attribute Related Caching Rules dialog box appears.

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

Note: By default, Oracle Web Cache provides definitions of two session identifiers that are commonly used by components of Oracle9i Application Server. The predefined session identifiers are:

JSESSIONID: Used for servlet session tracking. It conforms to the Java 2 Platform, Enterprise Edition (J2EE) standard. The cookie name is JSESSIONID; the embedded URL parameter is jsessionid.

FoundationPersistentSessionID: Used by Oracle9iAS Foundation Classes for persistent session tracking. The cookie name is ESFSID. There is no embedded URL parameter.

If the sessions listed do not contain the definition you require, then choose **Create Session/Personalized Attribute** 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/Attribute** 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. In the **Default Value (Optional)** field, enter a default string that Oracle Web Cache will use for the cookie or embedded URL parameter value. Oracle Web Cache uses the default string for those requests without the cookie or

parameter information. For these requests, Oracle Web Cache substitutes the personalized attribute or session ID information with the default string. The default string defaults to `default`.

Note: 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.

- d. Optionally, in the **Comment** field, enter a description of the definition.
 - e. Choose **Submit**.
6. For the prompt **1. Cache documents whose requests contain this session?** 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 prompt **2. Cache documents whose requests do not contain this session?**, select 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.
 8. If you answered **YES** to the prompts described in Steps 6 and 7, for the prompt **3. Can the document whose request doesn't contain this attribute be derived from the document whose request does contain this attribute by using the default value of the attribute?**, select either **YES** or **NO**:
 - Select **YES** to cache one version of the document. For those requests without a session cookie or embedded URL parameter, a default value is used.
 - Select **NO** to cache two different versions of the document. Oracle Web Cache serves one version to those requests that support session cookie or the embedded parameter and serves the other version to those requests that do not support the session cookie or embedded parameter.

9. In the Add Session/Personalized Attribute Related Caching Rule dialog box, choose **Submit**.
10. Associate the session-tracking rule with URLs:
 - a. In the Session/Personalized Attribute Related Caching Rules page, select the newly-created rule, and then choose **Change Selector Association**.
The Change Policy-Selector Association dialog box appears.
 - b. Select a selector from the right list, and then choose the **Make Association** button.
The selector moves to the left list and the dialog box closes. Proceed to Step 11.

If the selector you require does not exist, then create a cacheability rule for the pages the support session tracking, as described in "[Configuring Cacheability Rules](#)" on page 6-6. In Step 10 of the procedure, select **Apply the following** and a rule in the **Session/Personalized Attribute Related Caching** row of the Edit/Create Cacheability Rule dialog box.
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: Some Web sites require users to have sessions while surfing most pages. If you want to preserve the session requirement, then you need to create a session-related caching rule for those pages. This way, a request without a session will always be served by the application Web server.

For some popular site entry pages, such as "/", that typically require session establishment, session establishment effectively makes the page non-cacheable to all new users without a session. To cache these pages while preserving session establishment, make the following minor modifications to your application:

1. Create a blank page for the entry URL, such as "/", that redirects to the real entry page.
2. Make the application Web server create a session when the blank page is requested without session.
3. Create a cacheability rule for the real entry page and the blank page, as described in "[Configuring Cacheability Rules](#)" on page 6-6.

In Step 10 of the procedure, select **Apply the following** and then select a session-related caching rule with a value of **cache with session, no cache w/o session** in the **Session/Personalized Attribute Related Caching Rules** row of the Edit/Create Cacheability Rule dialog box.

In this way all initial user requests to the entry URL will first hit the blank page, which requires minimal resources to generate, and receive the response and session establishment from the application Web server. Subsequent redirected requests to the more expensive entry page will carry the session, enabling it to be served out of the cache.

Configuring Pages for Content Assembly and Partial Page Caching

See Also: ["Content Assembly and Partial Page Caching"](#) on page 2-17 for an overview of partial page caching

This section describes how to enable dynamic assembly of Web pages of HTML fragments and create rules for the cacheable and non-cacheable page fragments. It contains the following topics:

- [Enabling Partial Page Caching](#)
- [Using ESI for Simple Personalization](#)
- [Examples of ESI Usage](#)

Enabling Partial Page Caching

To enable partial page caching:

1. Create a template page and HTML fragments to include.
2. Configure the template page that the end user will request as follows:
 - a. Use ESI markup tags in the template to fetch and include the HTML fragments.

See Also: [Appendix D, "Edge Side Includes Language"](#)

Important: ESI tags cannot be used on a page that contains `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags. If you require simple personalization and are using ESI, see ["Using ESI for Simple Personalization"](#) on page 6-34.

- b. In the template page, use an HTTP `Surrogate-Control: content="ESI/1.0"` header in the HTTP response message:

`Surrogate-Control: max-age=30+60, content="ESI/1.0"`

See Also: ["Configuring Cacheability Attributes in Response Headers"](#) on page 6-45

- c. Create a cacheability rule for the template page, as described in ["Configuring Cacheability Rules"](#) on page 6-6.

In Step 9 of the procedure, select **Yes** or **No** to enable or disable other client caches to process ESI tags.

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

3. If the HTML fragments use ESI markup tags, use an HTTP `Surrogate-Control: content="ESI/1.0"` header in the HTTP response message.

See Also: ["Configuring Cacheability Attributes in Response Headers"](#) on page 6-45

4. Determine the cacheability of the HTML fragments, and then create cacheability rules for them.

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

Specify rules that cache the static HTML fragments and do not cache the truly dynamic fragments that require application Web server and database server generation.

Using ESI for Simple Personalization

You can use variable expressions to achieve the same substitution as personalized attribute and session-encoded URLs. Oracle Corporation recommends using ESI for simple personalization when you are utilizing other ESI features, otherwise continue to use the methods described in ["Configuring Rules for Pages with Simple Personalization"](#) on page 6-19.

For example, the following HTML excerpt uses the `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags to substitute a user's name based on the value the browser passes with `UserName` cookie. In addition, the session information contained within the `sessionID` cookie is used to replace session information for one user with another user.

```
Welcome <!-- WEBCACHETAG="UserName" -->John<!--WEBCACHEEND -->!  
Here is a <a href="/jsp/myPage.jsp?sessionID=13001">link</a>.
```

The same effect is achieved with the following ESI markup:

```
<esi:vars>
  Welcome ${HTTP_COOKIE{'username'}}!
  Here is a <a href="/jsp/myPage.jsp?sessionID=${QUERY_
STRING{'sessionid'}}">link</a>.
</esi:vars>
```

The `<esi:vars>` tag enables you to use an ESI environment variable outside of an ESI tag. Variables can also be used with other ESI tags.

See Also:

["Variable Expressions"](#) on page D-4

["ESI vars Tag"](#) on page D-18

Examples of ESI Usage

This section provides examples of ESI usage in the following topics:

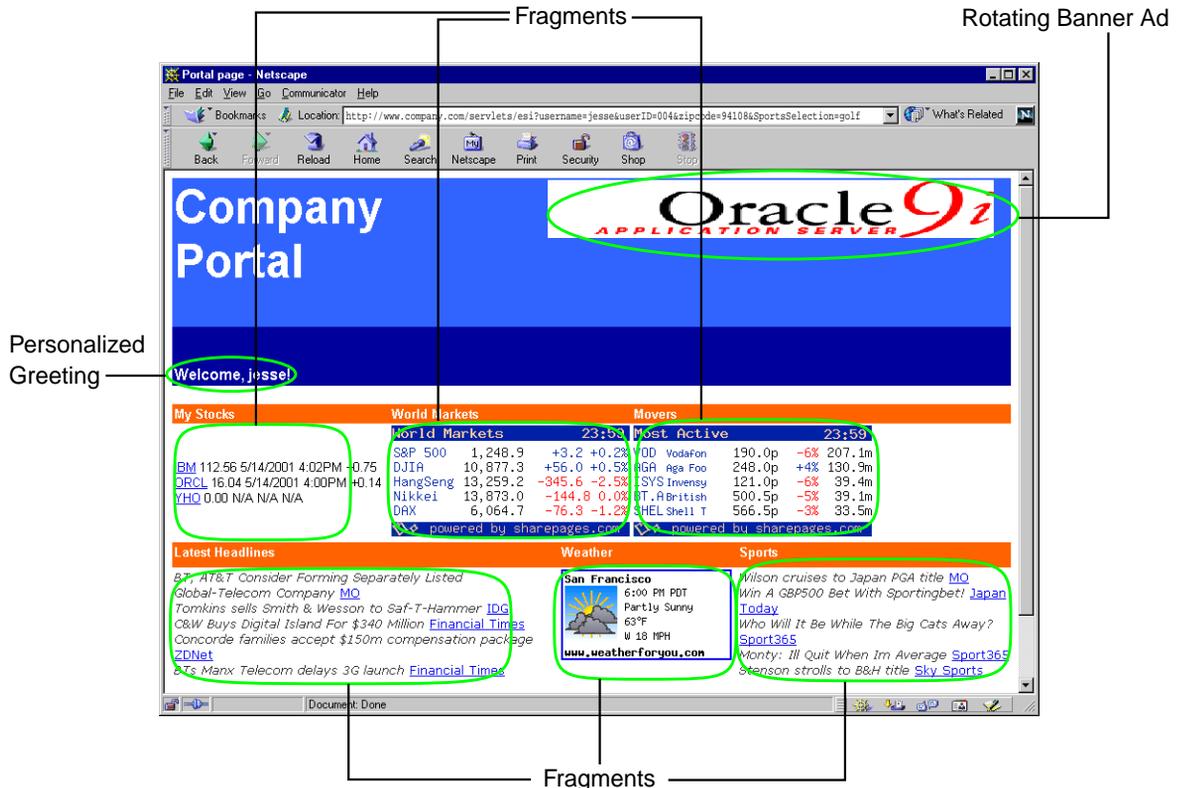
- [Example Portal Site Implementation](#)
- [Example of Simple Personalization with Variable Expressions](#)

Example Portal Site Implementation

Figure 6-8 shows a portal site page,

`http://www.company.com/servlets/esi?username=jesse&userID=004&zipcode=94108&SportsSelection=golf`, for a registered user named Jesse.

Figure 6-8 Portal Site Page



An esi Java servlet reads in the `portal.esi` file, which is a template page comprised of ESI markup tags, and processes the ESI tags to display the correct stock, news, and weather output for user Jesse. The servlet also sets the `Surrogate-Control: content="ESI/1.0"` header to enable ESI processing.

Without ESI, the servlet output would have been a pure HTML file. Because of its dynamic content, including a rotating banner ad, a personalized greeting, stock quotes, regional weather, and news content, this page would not be cacheable. For example, the `src` attribute of the `` tag for the banner ad changes for each request, making it uncacheable:

```

```

On the other hand, with ESI markup tags, Oracle Web Cache is able to assemble and cache most of the content. The rest of this section examines the ESI markup used in `portal.esi` and shows how ESI increases overall cacheability. Note that much of the HTML formatting has been removed for the sake of clarity.

Figure 6–9 shows the markup for the rotating ad banner and the personalized greeting `Welcome, jesse!`. The banner relies on alternate processing with the `<esi:try>` tag. If the servlet cannot run `AdBanner`, then a link to `www.oracle.com` appears in the banner's place. The personalized greeting is achieved by the `<esi:vars>` tag, which bases the greeting on the `username` parameter embedded in the URL. `username` is the registered user's name. This markup enables the banner and personalized greeting to be included in the cacheable template page.

Figure 6–9 *portal.esi Example: Rotating Banner and Personalized Greeting*

```
<table width="100%" cellpadding="2" cellspacing="0" border="0" align="Center"
height="98">
  <tbody>
    ....
    <!-- Rotating Ad Banner -->
      <esi:try>
        <esi:attempt>
          <esi:comment text="Include an ad banner <img> html tag"/>
          <esi:include src="/servlets/AdBanner"/>
        </esi:attempt>
        <esi:except>
          <esi:comment text="Just write some HTML instead"/>
          <a href=www.oracle.com>www.oracle.com</a>
        </esi:except>
      </esi:try>
    ....
    <!-- Personalized Greeting -->
      <p>
        <esi:vars>Welcome, ${QUERY_STRING{username}}!</esi:vars>
      </p>
    ....
  </tbody>
</table>
<br>
....
```

As shown in **Figure 6–10**, the response to the included image fragment for the banner is not cacheable. The first time a user requests this page, Oracle Web Cache sends the request to the application Web server to generate the banner. On the application Web server, `AdBanner` generates the banner for the request.

Figure 6–10 *portal.esi Example: Rotating Banner and Personalized Greeting Output*

```
<table width="100%" cellpadding="2" cellspacing="0" border="0" align="Center"
height="98">
  <tbody>
    ....
    <!-- Rotating Ad Banner -->
    <a href="www.companyad.com/redirect?item=11934502">
    </a>
    ....
    <!-- Personalized Greeting -->
      <p>
        Welcome, jesse!
      </p>

    </tbody>
  </table>
  <br>
  ....
```

As shown in [Figure 6–11](#), the next time the user reloads the page, AdBanner generates another banner for the request.

Figure 6–11 *portal.esi Example: Rotating Banner and Personalized Greeting Reload*

```
<table width="100%" cellpadding="2" cellspacing="0" border="0" align="Center"
height="98">
  <tbody>
    ....
    <!-- Rotating Ad Banner -->
    <a href="www.companyad.com/redirect?item=123456602">
    </a>
    ....
    <!-- Personalized Greeting -->
      <p>
        Welcome, jesse!
      </p>
    </tbody>
  </table>
  <br>
  ....
```

By separating the generation of the included image fragment response from the template page, Oracle Web Cache is able to cache the template and integrate the dynamic ad banner into the template.

The markup for My Stocks, World Markets, and Movers is depicted in [Figure 6–12](#). My Stocks includes a fragment named `PersonalizedStockSelection`. The displayed stocks are based on the `userID` parameter encoded in the URL. `userID` is the registered user’s unique ID. The World Markets and Movers sections rely on alternate processing with the `<esi:try>` tag. The `<esi:try>` tag first attempts to include data from external sites. If the content cannot be fetched, then the message `Sorry, can't display indexes` displays.

Figure 6–12 *portal.esi* Example: My Stocks, World Markets, and Movers Sections

```

....
<table width="100%" cellpadding="2" cellspacing="0" border="0">
  <tbody>
    <tr>
      <td>
        My Stocks
      </td>
      <td>
        World Markets
      </td>
      <td>
        Movers
      </td>
    </tr>
    <tr>
      <td width="26%">
        <!-- Personalized Stock Content -->
        <esi:include src="/servlets/PersonalizedStockSelection?userID=${QUERY_
STRING{userID}}" />
      </td>
      <td valign="Top" width="23%">

        <!-- External World Market Content -->
        <esi:try>
          <esi:attempt>
            <esi:include
src="/servlets/Proxy?url=http://www.service.com/WorldMarketIndex">
          </esi:attempt>
          <esi:except>
            Sorry, can't display indexes
          </esi:except>
        </esi:try>
      </td>

```

```

<!-- External Movers Content -->
<td valign="Top" width="51%">
<esi:try>
  <esi:attempt>
    <esi:include src="/servlets/Proxy?url=http://www.service.com/Movers">
  </esi:attempt>
  <esi:except>
    Sorry, can't display second set of indices
  </esi:except>
</esi:try>
</td>
</tr>
</tbody>
</table>
....

```

The markup for the included fragment `PersonalizedStockSelection` is depicted in [Figure 6-13](#). It includes fragments for three stock quotes: IBM, ORCL, and YHO.

Figure 6-13 *portal.esi Example: PersonalizedStockSelection Fragment for jesse*

```

<table width="160" cellspacing="0" cellpadding="0" border="0">
  <tbody>
    <tr>
      <br>
      <esi:include src="Quote?symbol=IBM"/>
      <br>
      <esi:include src="Quote?symbol=ORCL"/>
      <br>
      <esi:include src="Quote?symbol=YHO"/>
      <br>
    </tr>
  </tbody>
</table>

```

Because the output is different for each user, the `PersonalizedStockSelection` fragment is not cacheable. However, the response to each of the included quotes is cacheable, enabling stock quotes to be shared by multiple users. Even when many users share quotes, only one browser reload is needed when the quotes are updated. For example, the `PersonalizedStockSelection` fragment for another user named Scott is depicted in [Figure 6-14](#). It includes fragments for three stock quotes: IBM, ORCL, and SCO. As already described, IBM and ORCL are also shared by Jesse. If Jesse reloads the page first and caches the quotes, then the IBM and ORCL quotes for Scott are automatically refreshed.

Figure 6-14 *portal.esi Example: PersonalizedStockSelection Fragment for scott*

```
<table width="160" cellspacing="0" cellpadding="0" border="0">
  <tbody>
    <tr>
      <br>
      <esi:include src="Quote?symbol=IBM"/>
      <br>
      <esi:include src="Quote?symbol=ORCL"/>
      <br>
      <esi:include src="Quote?symbol=SCO"/>
      <br>
    </tr>
  </tbody>
</table>
```

Figure 6–15 shows the markup for Latest Headlines, Weather, and Sports.

Latest Headlines displays the content based on the user's `NewsSelection` category, `internet` or `finance`, by using conditional processing with the `<esi:choose>` tag. Because the URL of the Company Portal page in Figure 6–8 on page 6-36 did not embed the `NewsSelection` parameter, the output includes the top news stories, `/News?type=Top&topic=TopNews`.

Weather uses the `<esi:vars>` tag to display the current weather conditions based on the registered user's zip code. Because Jesse entered in `zipcode=94108` in the URL, the output includes weather conditions for San Francisco, California.

Sports displays the sports content based on the user's `SportsSelection` category: `golf`, `soccer`, `football`, `baseball`, or `basketball`. Because the URL for user Jesse embeds `SportsSelection=golf`, the output includes headlines relating to golf, `/News?type=Top&topic=GolfNews`.

Figure 6–15 portal.esi Example: Headline News, Weather, and Sports Sections

```

....
<!-- News, Weather, and Sports Content -->
<table width="100%" cellpadding="2" cellspacing="0" border="0">
  <tbody>
    <tr>
      <td>
        Latest Headlines
      </td>
      <td>
        Weather
      </td>
      <td>
        Sports
      </td>
    </tr>
    <tr>
      <td>
        <!-- News Content -->
        <!-- Use the esi include tag so that different portal users can share the
        included news HTML snippet -->
        <esi:choose>
          <esi:when test="$(QUERY_STRING{NewsSelection}) == 'internet'">
            <esi:include src="/News?type=Top&topic=TopTechNews"
onerror="continue"/>
          </esi:when>
          <esi:when test="$(QUERY_STRING{NewsSelection}) == 'finance'">
            <esi:include src="/News?type=Top&topic=TopFinanceNews"
onerror="continue"/>

```

```

        </esi:when>
        <esi:otherwise>
            <esi:include src="/News?type=Top&topic=TopNews" onerror="continue"/>
        </esi:otherwise>
    </esi:choose>
</td>
<td>
    <!-- Weather -->
    <esi:vars>
        
    </esi:vars>
</td>
<!-- Sports -->
<!-- Use the esi include tag so that different portal users can share the
included sports HTML snippet -->
    <esi:choose>
        <esi:when test="${QUERY_STRING{SportsSelection}} == 'golf'">
            <esi:include src="/News?type=Top&topic=GolfNews" onerror="continue"/>
        </esi:when>
        <esi:when test="${QUERY_STRING{SportsSelection}} == 'soccer'">
            <esi:include src="/News?type=Top&topic=SoccerNews" onerror="continue"/>
        </esi:when>
        <esi:when test="${QUERY_STRING{SportsSelection}} == 'football'">
            <esi:include src="/News?type=Top&topic=FootballNews"
onerror="continue"/>
        </esi:when>
        <esi:when test="${QUERY_STRING{SportsSelection}} == 'baseball'">
            <esi:include src="/News?type=Top&topic=BaseballNews"
onerror="continue"/>
        </esi:when>
        <esi:when test="${QUERY_STRING{SportsSelection}} == 'basketball'">
            <esi:include src="/News?type=Top&topic=BasketballNews"
onerror="continue"/>
        </esi:when>
    <esi:otherwise>
        <esi:include src="/News?type=Top&topic=SportsNews" onerror="continue"/>
    </esi:otherwise>
    </esi:choose>
</td>
</tr>
</tbody>
</table>

```

Example of Simple Personalization with Variable Expressions

As described in Step 3 of "[Configuring Rules for Pages with Simple Personalization](#)" on page 6-19, the `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags can be used between other HTML tag pairs, but not within an HTML tag. However, ESI variables can be used within an HTML tag.

For example, consider [Figure 6-16](#). Its HTML code uses PL/SQL for an HTML form with a text box in it.

Figure 6-16 PL/SQL Code without Personalization

```
http.p('<form action="test" method="GET">');
http.p('<table border="0" >
  <tr>
    <td><input type="text" name="p_name" size="8" value="'|p_name|'|"></td>
  </tr>
  <tr>
    <td><input type="submit" value="Search"></td>
  </tr>
</table>');
```

[Figure 6-17](#) shows how the `$HTTP_COOKIE` variable is used with the `<esi:vars>` tag to replace the value of `p_name` with the user's name.

Figure 6-17 PL/SQL Code with Personalization through ESI

```
http.p('<form action="test" method="GET">');
http.p('<table border="0" >
  <tr><esi:vars>
    <td><input type="text" name="p_name" size="8" value="$ (HTTP_
COOKIE{'p_name'})"></td>
  </tr></esi:vars>
  <tr>
    <td><input type="submit" value="Search"></td>
  </tr>
</table>');
```

Configuring Cacheability Attributes in Response Headers

In addition to creating cacheability rules with Oracle Web Cache Manager interface, application developers can choose to store some of the cacheability attributes in the header of an HTTP response message. This feature enables the application Web server to override the settings configured through the Oracle Web Cache Manager interface, as well as allowing other third-party caches to use Oracle Web Cache cacheability attributes.

To enable this feature, configure the HTTP response with the `Surrogate-Control` response-header field as follows:

```
Surrogate-Control: control_directive, control_directive, ...
```

Table 6–4 describes the supported control directives.

Table 6–4 *Surrogate-Control Control Directives*

Control Directive	Description
<code>no-store</code>	Specify for Oracle Web Cache not to cache the document.
<code>no-store-remote</code>	Specify to now allow other ESI-compliant caches, such as third-party Content Delivery Network (CDN), to cache the document. The <code>no-store-remote</code> directive has similar semantics to the <code>no-store</code> directive, except that it is only be honored by "remote" caches. Generally, this means those caches that are more than one or two hops from the application Web server, such as caches in a Content Delivery Network (CDN). This directive is especially useful if you want to cache volatile content locally, where invalidation propagation is immediate, but not in a distributed network of upstream ESI processors, where invalidation may take several minutes.
<code>max-age</code>	Specify to enable Oracle Web Cache to cache the document. Specify the time, in seconds, to expire the document. Optionally, specify the time, in seconds, to remove the document from the cache after the expiration time. Use the following format: <code>max-age=expiration_time [+ removal_time]</code> The default removal time is 0 seconds.

Control Directive	Description
content	<p>Specify what kind of processing is required:</p> <ul style="list-style-type: none"> ■ "ESI/1.0" to process ESI tags for content assembly and partial page caching ■ "webcache/1.0" to process the <!-- WEBCACHETAG--> and <!-- WEBCACHEEND--> tags for personalized attributes and session-encoded URLs

Usage Notes

- Control directives are case sensitive.
- no-store and no-store-remote are mutually exclusive.
- content="ESI/1.0" and content="webcache/1.0" are mutually exclusive.

Example Usage

In the following example, the `Surrogate-Control` response-header field specifies that the document is to expire in 30 seconds and be removed 60 seconds after expiration. It also specifies that the document contains ESI tags that require processing:

```
Surrogate-Control: max-age=30+60, content="ESI/1.0"
```

In the following example, the `Surrogate-Control` response-header field specifies that the document is not to be cached:

```
Surrogate-Control: no-store
```

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 and Failover](#)
- [Binding a Session to an Application Web Server](#)

Configuring Load Balancing and Failover

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 1-10 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 3: Specify Web Site Settings"](#) on page 5-5. 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 a failover threshold has been met.

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 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.

See Also: ["Task 3: Specify Web Site Settings"](#) on page 5-5 to configure the failover threshold

Binding a Session to an Application Web Server

See Also: ["Application Web Server Binding"](#) on page 1-14 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: Oracle Web Cache requires a session cookie to perform session binding.

If browsers do not support cookies and you want to use an embedded URL parameter for the session, then perform the following in order for Oracle Web Cache to perform session binding on the session:

1. In addition to the **URL Parameter** field, specify a cookie name for the session in the **Cookie Name** field.
2. Ensure that your application Web server returns a `Set-Cookie` response-header with the value of the session every time a session is created.

```
Set-Cookie: cookie=value
```

Set *value* to the same value as set in the **URL Parameter** field.

Oracle Web Cache uses the `Set-Cookie` response header, even if ignored by browsers, to locate the session cookie value for session binding.

See Also: <http://rfc.net/rfc2965.html> for further information about the `Set-Cookie` response header

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.

This chapter 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, you must stop and restart Oracle Web Cache. To start, stop, or restart Oracle Web Cache, use either Oracle Web Cache Manager or the `webcachectl` utility.

When you stop Oracle Web Cache, all objects are cleared from the cache. In addition, all statistics are cleared.

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. 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 <code>cache</code> 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 4-2 2. 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, then Oracle Web Cache is not running. Continue to Step 2. <code>Oracle Web Cache admin server is NOT running.</code> <code>Oracle Web Cache cache server is NOT running.</code> If the following message appears, then Oracle Web Cache is already running. <code>Oracle Web Cache admin server is running (pid=pid).</code> <code>Oracle Web Cache cache server is running (pid=pid).</code> 2. Start Oracle Web Cache. From the command line, enter: <code>webcachectl start</code> The following message appears: <code>Oracle Web Cache started</code>
	To stop Oracle Web Cache, from the command line, enter:
	<pre>webcachectl stop</pre> <p>The following message appears:</p> <pre>Oracle Web Cache admin server stopping.</pre> <pre>Oracle Web Cache cache server stopping.</pre>

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

1. Select the **Services** icon in the Control Panel window.
The Services window appears.
2. Select the `OracleHOME_NAMEWebCacheAdmin` service to start the admin server, and then choose **Start** to start the service.
3. Select the `OracleHOME_NAMEWebCache` service to start the cache server, and then choose **Start** to start the service.
4. 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 on HTTP.

To change the default port number or protocol:

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 **Invalidation Port** field, enter the new port.
- d. From the **Protocol** list, select either **HTTP** or **HTTPS** to accept invalidation requests from one of the following URLs:

http://web_cache_hostname:http_port

https://web_cache_hostname:https_port

Note: If you enable HTTPS for invalidation requests, you cannot use the Oracle Web Cache Manager interface to send the requests. You must submit requests through the URL.

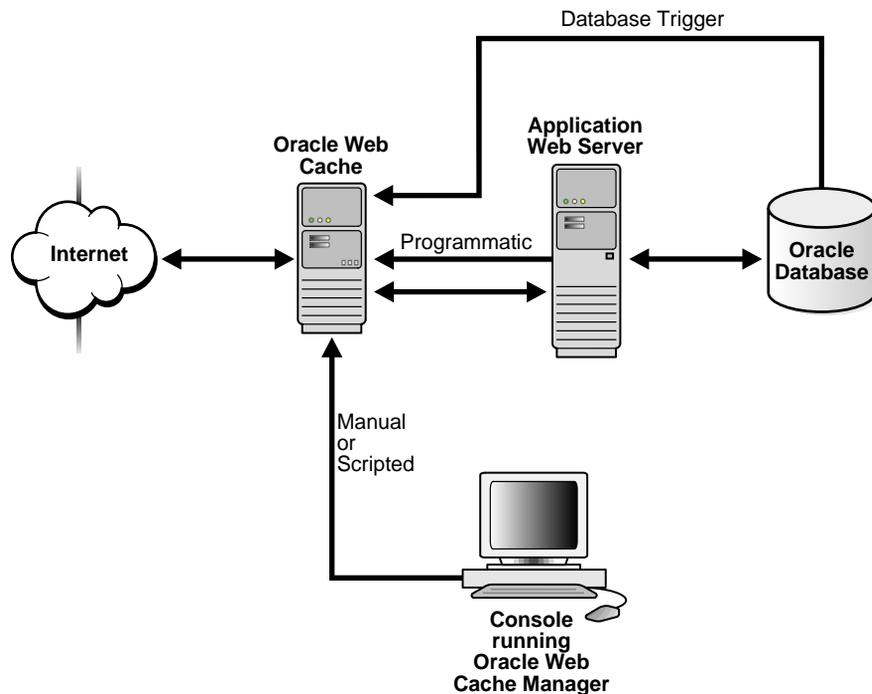
- e. Choose **Submit**.
3. Apply changes and restart Oracle Web Cache:
 - a. In the Oracle Web Cache Manager main window, choose **Apply Changes**.
 - b. In the navigator pane, select **Administering Oracle Web Cache > Web Cache Operations**.
The Oracle Web Cache Operations page appears in the right 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` messages written in **Extensible Markup Language (XML)** syntax. The contents of the XML message body tell 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 an 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 web_cache_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 encoding of invalidator:invalidator_password>
content-length: #bytes
```

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

```
Authorization: BASIC aW52YWxpZGF0b3I6YWRtaW4=
```

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

See Also:

- <http://rfc.net/rfc1421.html> for information about password base64 encoding
- ["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 XML syntax.

Invalidation Message

Note: Oracle Web Cache continues to support invalidation messages sent in the following release 1.0 format:

```
<?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>
```

Use the following syntax to invalidate document(s) contained within an exact URL that includes the complete path and file name:

```
<?xml version="1.0" ?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.0">
  <OBJECT>
    <BASICSELECTOR URI="URL" />
    <ACTION REMOVALTTL="TTL" />
  </OBJECT>
</INVALIDATION>
```

Use the following syntax to invalidate document(s) based on more advanced invalidation selectors:

```
<?xml version="1.0" ?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.0">
  <OBJECT>
    <ADVANCEDSELECTOR URIPREFIX="prefix"
      URIEXP="URL_expression"
      METHOD="HTTP_request_method">
      <COOKIE NAME="cookie_name" VALUE="value"/>
      <HEADER NAME="HTTP_request_header" VALUE="value"/>
    </ADVANCEDSELECTOR>
    <ACTION REMOVALTTL="TTL"/>
  </OBJECT>
</INVALIDATION>
```

[Table 8-1](#) describes the invalidation message elements and attributes.

Table 8-1 Invalidation Message Syntax

Invalidation Element/Attribute	Description
OBJECT	Required element in the invalidation message. You can specify more than one OBJECT element in the message.
BASICSELECTOR	URI: Required attribute of the BASICSELECTOR element. Specify the URL of the document(s) to be invalidated. Include the complete path and file name.

Invalidation Element/Attribute	Description
ADVANCEDSELECTOR	<ul style="list-style-type: none"> <li data-bbox="441 322 1308 574"> <p>■ URIPREFIX</p> <p>Required attribute for the ADVANCEDSELECTOR element.</p> <p>Specify the prefix path (beginning and ending with "/") of the document(s) to be invalidated.</p> <p>The prefix is interpreted literally, including reserved regular expression characters. Reserved regular expression characters include periods (.), question marks (?), asterisks (*), brackets ([]), curly braces ({ }), carets (^), dollar signs (\$), and backslashes (\).</p> <li data-bbox="441 591 1308 817"> <p>■ URIEXP</p> <p>Optional attribute for the ADVANCEDSELECTOR element.</p> <p>Specify the URL of the document(s) to be invalidated underneath the URIPREFIX. If no value is entered, then everything under the URIPREFIX will be matched.</p> <p>Regular expression characters are permitted. To interpret these characters literally, escape them with a backslash (\).</p> <li data-bbox="441 835 1308 956"> <p>■ METHOD</p> <p>Optional attribute for the ADVANCEDSELECTOR element.</p> <p>Specify HTTP request method (GET or POST) of the document(s) to be invalidated.</p> <li data-bbox="441 973 1308 1107"> <p>■ BODYEXP</p> <p>Optional attribute for the ADVANCEDSELECTOR element.</p> <p>If the METHOD is POST, specify HTTP POST body message of the document(s) to be invalidated.</p>

Invalidation Element/Attribute	Description
	<ul style="list-style-type: none"> <p>■ <code>COOKIE</code></p> <p>Optional element in the invalidation message.</p> <p>NAME: Required attribute for the <code>COOKIE</code> element. Specify the cookie name to invalidate documents based on the cookie. The name must match a cookie name associated with a cacheability rule or session/personalized attribute caching rule for the URL.</p> <p>VALUE: Optional attribute for the <code>COOKIE</code> element.</p> <p>Specify the value of the cookie. If no value is present, then only documents with the named cookie but without value are invalidated.</p> <p>■ <code>HEADER</code></p> <p>Optional element in the invalidation message.</p> <p>NAME: Required attribute for the <code>HEADER</code> element. Specify the HTTP request header and its value to invalidate based on the request header. The header must match a header associated with a cacheability rule for the URL.</p> <p>VALUE: Optional attribute for the <code>HEADER</code> element. Specify the value of the header.</p>
<code>ACTION</code>	Required element in the invalidation message
<code>REMOVALTTL</code>	<p>Optional attribute for the <code>ACTION</code> element</p> <p>Specify the maximum time that documents can reside in the cache before they are invalidated. The default is 0 seconds.</p>

Note: The following special XML characters must be escaped in the `URI`, `URIPREFIX`, and `URIEXP` fields: ampersand (&) with "& ;", greater than sign (>) with ">", less than sign (<) with "<", double quotes (") with """, and single quotes (') with "&apos ;".

See Also: ["Invalidation Request DTD"](#) on page C-2 for further information about invalidation request syntax

Invalidation Response

Invalidation responses are returned in the following format for BASICSELECTOR invalidation messages:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONRESULT VERSION="WCS-1.0">
  <OBJECTRESULT>
    <BASICSELECTOR URI="URL">
    </BASICSELECTOR>
    <RESULT ID="ID" STATUS="status" NUMINV="number"/>
  </OBJECTRESULT>
</INVALIDATIONRESULT>
```

Invalidation responses are returned in the following format for ADVANCEDSELECTOR invalidation messages:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONRESULT VERSION="WCS-1.0">
  <OBJECTRESULT>
    <ADVANCEDSELECTOR URIPREFIX="prefix"
      URIEXP="URL_expression">
      METHOD="HTTP_request_method">
    <COOKIE NAME="cookie_name" VALUE="value"/>
    <HEADER NAME="HTTP_request_header" VALUE="value"/>
  </ADVANCEDSELECTOR>
  <RESULT ID="ID" STATUS="status" NUMINV="number"/>
</OBJECTRESULT>
</INVALIDATIONRESULT>
```

[Table 8–2](#) describes the invalidation response syntax. Note that `BASICSELECTOR` and `ADVANCEDSELECTOR` are described in [Table 8–1](#) on page 8-8.

Table 8–2 Invalidation Response Syntax

Invalidation Element/Attribute	Description
RESULT	Invalidation result
ID	Sequence number of all the invalidation objects sent in the invalidation response. If there are multiple selectors specified in the invalidation message, then the sequence number starts at 1 for the first URL and continues sequentially for each additional selector.
STATUS	Status of the invalidation. Status can be one of the following: <ul style="list-style-type: none"> ▪ <code>SUCCESS</code> for successful invalidations ▪ <code>URI NOT CACHEABLE</code> for documents that are not cacheable ▪ <code>URI NOT FOUND</code> 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. 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. Specify which documents to invalidate:

Note: When using Oracle Web Cache Manager, the following characters are permitted in the **Enter exact URL for removal**, **URL Path Prefix**, and **URL Regular Expression** fields: ampersand (&), greater than sign (>), less than sign (<), double quotes ("), and single quotes (').

Basic Invalidation

**Remove all
cached
documents.**

Select to remove all documents from the cache.

**Enter exact
URL for
removal**

Specify the URL of the document(s) to be invalidated. Include the complete path and file name.

Advanced Invalidation

**URL Path
Prefix**

Required. Specify the prefix path (beginning and ending with "/") of the document(s) to be invalidated.

The prefix is interpreted literally, including reserved regular expression characters. These characters include periods (.), question marks (?), asterisks (*), brackets ([]), curly braces ({ }), carets (^), dollar signs (\$), and backslashes (\).

**URL Regular
Expression**

Optional. Specify the URL of the document(s) to be invalidated underneath the **URL Path Prefix**. If no value is entered, then everything under the **URL Path Prefix** will be matched.

Regular expression characters are permitted. To interpret these characters literally, escape them with a backslash (\).

HTTP Method	<i>Optional.</i> Select the HTTP request method (GET or POST) of the document(s) to be invalidated.
POST Body Expression	<i>Optional.</i> If POST is selected for the HTTP Method , enter the HTTP body message of the document(s) to be invalidated.
Cookie Information	<p><i>Optional.</i> On UNIX, select the cookie name for the document(s) to be invalidated from the list, and enter its value in the Value field.</p> <p>On Windows, enter the cookie name for the document(s) to be invalidated in the Name field, and enter its value in the Value field.</p>
Header Information	<i>Optional.</i> Enter the HTTP request header for the document(s) to be invalidated in the Name field, and enter its value in the Value field. The name must match the header associated with a cacheability rule associated for the URL.

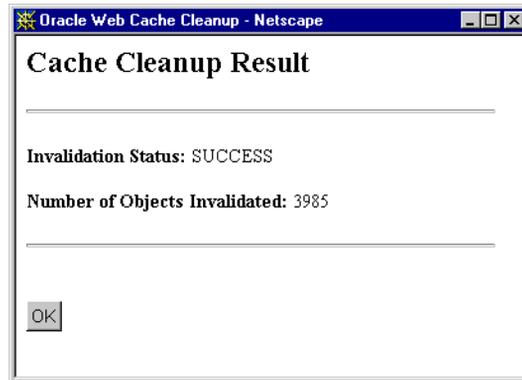
- In the **Action** section, specify how to process invalid documents.

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.
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.

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.

- Choose **Submit**.

Oracle Web Cache processes the invalidation request, and returns a Cache Cleanup dialog box that shows the invalidation status and the number of objects invalidated. For example:



For prefix-based invalidations that require Oracle Web Cache to traverse a complex directory structure, invalidation can take some time. Therefore, do not choose Submit again until the Cache Cleanup Result dialog box appears. Creating a queue of invalidation requests can degrade the performance of Oracle Web Cache.

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 do this, use the `UTL_TCP` Oracle supplied package to send invalidation messages through database triggers.

See Also:

- `readme.examples.html` in the `$ORACLE_HOME/webcache/examples` directory on UNIX and `ORACLE_HOME\webcache\examples` directory on Windows 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
- Oracle PL/SQL documentation

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 or C source that you can link with your applications for generating invalidation messages.

See Also: `readme.examples.html` in the `$ORACLE_HOME/webcache/examples` directory on UNIX and `ORACLE_HOME\webcache\examples` directory on Windows for further information about using the `Invalidate.java` or `invalidate.c` file

Invalidation Examples

This section contains the following invalidation message examples:

- [Example: Invalidating One Document](#)
- [Example: Invalidating Multiple Objects](#)
- [Example: Invalidating a Subtree of Documents](#)
- [Example: Invalidating All Documents for a Web Site](#)
- [Example: Invalidating Documents with the Prefix](#)

The examples in this section 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

The following message invalidate file `/images/logo.gif`:

```
<?xml version="1.0" ?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.0">
  <OBJECT>
    <BASICSELECTOR URI="/images/logo.gif"/>
    <ACTION/>
  </OBJECT>
</INVALIDATION>
```

Invalidation response:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONRESULT VERSION="WCS-1.0">
  <OBJECTRESULT>
    <BASICSELECTOR URI="/images/logo.gif"/>
    <RESULT ID="1" STATUS="SUCCESS" NUMINV="1"/>
  </OBJECTRESULT>
</INVALIDATIONRESULT>
```

The following message invalidates a document exactly matching /contacts/contacts.html using the BASICSELECTOR element:

```
<?xml version="1.0" ?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.0">
  <OBJECT>
    <BASICSELECTOR URI="/contacts/contacts.html"/>
    <ACTION REMOVALTTL="0"/>
  </OBJECT>
</INVALIDATION>
```

This is equivalent to the following using the ADVANCEDSELECTOR element:

```
<?xml version="1.0" ?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.0">
  <OBJECT>
    <ADVANCEDSELECTOR URIPREFIX="/contacts/"
      URIEXP="^/contacts/contacts\\.html$"/>
    <ACTION REMOVALTTL="0"/>
  </OBJECT>
</INVALIDATION>
```

The ADVANCEDSELECTOR element uses the URIPREFIX attribute. This attribute is used to traverse the directory structure. The quicker invalidation reaches the right tree level, the quicker the invalidation process is done. The message with the BASICSELECTOR element is the more efficient of the two examples because there is no directory structure traversal involved.

Example: Invalidating Multiple Objects

The following message invalidates two different objects, `summary.jsp` and `summary.gif`:

```
<?xml version="1.0" ?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.0">
  <OBJECT>
    <ADVANCEDSELECTOR URIPREFIX="/global/sales/"
                      URIEXP="summary.jsp\?year=2001">
      <COOKIE NAME="group" VALUE="asia" />
    </ADVANCEDSELECTOR>
    <ACTION />
  </OBJECT>
  <OBJECT>
    <ADVANCEDSELECTOR URIPREFIX="/image/"
                      URIEXP="summary.*\.gif$" />
    <ACTION />
  </OBJECT>
</INVALIDATION>
```

Invalidation response:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONRESULT VERSION="WCS-1.0">
  <OBJECTRESULT>
    <ADVANCEDSELECTOR URIPREFIX="/global/sales/"
                      URIEXP="summary.jsp\?year=2001" >
      <COOKIE NAME="group" VALUE="asia" />
    </ADVANCEDSELECTOR>
    <RESULT ID="1" STATUS="SUCCESS" NUMINV="2"/>
  </OBJECTRESULT>
  <OBJECTRESULT>
    <ADVANCEDSELECTOR URIPREFIX="/image/" URIEXP="summary.*\.gif$" >
    </ADVANCEDSELECTOR>
    <RESULT ID="2" STATUS="SUCCESS" NUMINV="14"/>
  </OBJECTRESULT>
</INVALIDATIONRESULT>
```

Example: Invalidating a Subtree of Documents

The following message invalidates all documents under the `/images/` directory:

```
<?xml version="1.0" ?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.0">
  <OBJECT>
    <ADVANCEDSELECTOR URIPREFIX="/images/" />
    <ACTION REMOVALTTL="0" />
  </OBJECT>
</INVALIDATION>
```

Invalidation response:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONRESULT VERSION="WCS-1.0">
  <OBJECTRESULT>
    <ADVANCEDSELECTOR URIPREFIX="/images/" />
    <RESULT ID="1" STATUS="SUCCESS" NUMINV="125" />
  </OBJECTRESULT>
</INVALIDATIONRESULT>
```

The following message invalidates all documents under `/contacts/` directory whose file names ends in `.html` and uses cookie name `cust` with a value of `oracle`:

```
<?xml version="1.0" ?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.0">
  <OBJECT>
    <ADVANCEDSELECTOR URIPREFIX="/contacts/"
                      URIEXP=".html$" />
    <COOKIE NAME="cust" VALUE="oracle" />
  </ADVANCEDSELECTOR>
  <ACTION REMOVALTTL="0" />
</OBJECT>
</INVALIDATION>
```

Invalidation response:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONRESULT VERSION="WCS-1.0">
  <OBJECTRESULT>
    <ADVANCEDSELECTOR URIPREFIX="/contacts"/>
      URIEXP=".html$"
    <COOKIE NAME="cust" VALUE="oracle"/>
  </ADVANCEDSELECTOR>
  <RESULT ID="1" STATUS="SUCCESS" NUMINV="45"/>
</OBJECTRESULT>
</INVALIDATIONRESULT>
```

Example: Invalidating All Documents for a Web Site

The following message invalidates all documents under /.

```
<?xml version="1.0" ?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.0">
  <OBJECT>
    <ADVANCEDSELECTOR URIPREFIX="/"/>
    <ACTION REMOVALTTL="0"/>
  </OBJECT>
</INVALIDATION>
```

Invalidation response:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONRESULT VERSION="WCS-1.0">
  <OBJECTRESULT>
    <ADVANCEDSELECTOR URIPREFIX="/"/>
    <RESULT ID="1" STATUS="SUCCESS" NUMINV="17"/>
  </OBJECTRESULT>
</INVALIDATIONRESULT>
```

Example: Invalidating Documents with the Prefix

To better understand the relationship of the URIPREFIX and URIEXP attributes, consider the examples that follow.

To invalidate `sample.gif` files within `/cec/cstage/graphic*` directories, use the following syntax:

```
<ADVANCEDSELECTOR URIPREFIX="/cec/cstage/"
  URIEXP="graphic.*/sample\.gif">
</ADVANCEDSELECTOR>
```

Note that `"."` in `"graphic.*/sample\.gif"` are regular expression characters that match all directories starting with `graphic`. The `"."` in `"sample\.gif"` is escaped for a literal interpretation.

If you used the following syntax, Oracle Web Cache would try to locate a directory named `graphic*`:

```
<ADVANCEDSELECTOR URIPREFIX="/cec/cstage/graphic*/"
  URIEXP="sample\.gif">
</ADVANCEDSELECTOR>
```

To invalidate documents contained within `/cec/cstage?ecaction=viewitem`, use the following syntax:

```
<ADVANCEDSELECTOR URIPREFIX="/cec/"
  URIEXP="cstage\?ecaction=viewitem">
</ADVANCEDSELECTOR>
```

Note that `"?"` is escaped with a backslash.

URLs such as `/cec/cstage?ecaction=viewitem&zip=94405` and `/cec/cstage?ecaction=viewitem&zip=94305` match and are invalidated, but `/usa/cec/cstage?ecaction=viewitem&zip=94209` does not match and is not invalidated.

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.

See Also: [Appendix E, "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

The following shows an event log excerpt with successful startup entries:

```
14/May/2001:11:34:15 -0800 -- Information: Maximum number of file/socket
descriptors set to 950.
14/May/2001:18:34:15 +0000 -- Information: Maximum connections possible are 900
14/May/2001:18:34:15 +0000 -- Information: Listening on ADMINISTRATION port 4000
address 0.0.0.0
14/May/2001:18:34:15 +0000 -- Information: The admin server started successfully
14/May/2001:11:34:15 -0800 -- Information: Maximum number of file/socket
descriptors set to 950.
14/May/2001:18:34:15 +0000 -- Information: Maximum connections possible are 900
14/May/2001:18:34:18 +0000 -- Information: Listening on NORM port 1100 address
0.0.0.0
14/May/2001:18:34:18 +0000 -- Information: Listening on INVALIDATION port 4001
address 0.0.0.0
14/May/2001:18:34:18 +0000 -- Information: Listening on STATISTICS port 4002
address 0.0.0.0
14/May/2001:18:34:21 +0000 -- Information: The cache server started successfully
14/May/2001:18:34:22 +0000 -- Information: The cache server is started by the
admin server at startup
```

Example: Event Log with Unsuccessful Startup Entries

The following 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.

```
14/May/2001:16:37:41 -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/May/2001:16:37:41 -0800 -- Error: Failed to start the server.
14/May/2001:16:37:41 -0800 -- Error: The server could not initialize
14/May/2001:16:37:41 -0800 -- Information: The server is exiting
```

Example: Event Log with an Invalidation Entry

The following shows an event log excerpt with an event associated with an invalidation request for the removal of document `personal.htm`:

```
14/May/2001:22:52:52 -0500 -- Information: <Invalidation>1 URLs with prefix
personal.htm have been successfully invalidated.
```

Example: Event Log with an Invalidation Message Error

The following shows an event log excerpt with an XML invalidation message error. In this example, Oracle Web Cache is unable to parse the message:

```
Example: Errors in the XML parsing. Note: The configuration files and
invalidation files use XML
files.
14/May/2001:10:55:26 -0500 -- Error: Invalidation XML Buffer cannot be parsed.
14/May/2001:10:55:26 -0500 -- Error: XML parsing error.
```

Example: Event Log with Shutdown Entries

The following shows an event log excerpt with typical shutdown entries:

```
14/May/2001:11:16:55 -0700 -- Information: SIGTERM caught - program will shut
down once all connections are complete.
14/May/2001:11:16:55 -0800 -- Information: SIGTERM caught - program will shut
down once all connections are complete.
14/May/2001:11:16:55 -0700 -- Information: The server is exiting
14/May/2001:11:16:55 -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 errors by the current day, enter `grep " Error:" event_log | grep "dd/mon/yyyy"`. For example:

```
grep " Error:" event_log | grep "19/May/2001"
```

To list errors by the current day and hour, enter `grep " Error:" event_log | grep "dd/mon/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. 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 **Extended LogFile Format (XLF)**, which is a superset of the **Common LogFile Format (CLF)**. Table 8–3 lists the XLF fields you can enter.

Table 8–3 XLF Fields for Access Logs

Field	Description
<code>cs(User-Agent)</code>	Information about the user agent originating the request
<code>cs(Referer)</code>	Allows the client to specify the address (URI) of the resource from which the Request-URI was obtained
<code>c-auth-id</code>	Username if the request contained an attempt to authenticate
<code>bytes</code>	Content length of the transferred document
<code>date</code>	Date at which the transaction completed
<code>time</code>	Time at which the transaction completed
<code>time-end</code>	Time at which the transaction ended
<code>time-start</code>	Time at which the transaction started
<code>time-taken</code>	Amount of time taken (in seconds) for transaction to complete
<code>c-ip</code>	Client's IP address and port
<code>s-ip</code>	Oracle Web Cache's IP address and port
<code>sc-method</code>	Oracle Web Cache-to-client HTTP request method (GET, POST, or others)

Field	Description
<code>sc-status</code>	<p>Oracle Web Cache-to-client HTTP status code:</p> <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. <p>See Also: http://rfc.net/rfc2616.html for further information about HTTP status codes</p>
<code>cs-uri</code>	Client-to-Oracle Web Cache URI
<code>cs-uri_stem</code>	Client-to-Oracle Web Cache stem portion of URI, omitting the query
<code>cs-uri-query</code>	Client-to-Oracle Web Cache query portion of URI, omitting the stem
<code>prefix(header)</code>	<p><i>header</i> is an HTTP header field and <i>prefix</i> is one of the following:</p> <p>cs: Client-to-Oracle Web Cache</p> <p>sc: Oracle Web Cache-to-client</p>

The order in which fields are entered determines the order in which the fields are logged.

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
```

Note that entered XLF fields are not added to the CLF default. If you want to use the CLF fields in addition to XLF fields, you must enter them.

Access Log Examples

The access log that follows uses 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.

```
138.2.213.146 - - [19/May/2001:10:27:42 -0500] "GET /~ssandrew/personal.htm
HTTP/1.0" 200 2438
138.2.213.146 - - [19/May/2001:10:27:54 -0500] "GET
/~ssandrew/personal.htm?UserName=Bob HTTP/1.0"
200 2438
138.2.213.146 - - [19/May/2001:10:47:30 -0500] "GET /~ssandrew/count.sh
HTTP/1.0" 403 289
138.2.213.146 - - [19/May/2001:10:47:34 -0500] "GET /~ssandrew/sbin/count.sh
HTTP/1.0" 200 321
138.2.213.146 - - [19/May/2001:10:47:41 -0500] "GET /sbin/count.sh HTTP/1.0" 200
321
138.2.213.146 - - [19/May/2001:11:34:23 -0500] "GET /cache.htm HTTP/1.0" 200 250
138.2.213.146 - - [19/May/2001:11:38:23 -0500] "GET /cache.htm HTTP/1.0" 304 0
138.2.213.146 - - [19/May/2001:11:38:48 -0500] "GET /cache.htm HTTP/1.0" 304 0
206.223.27.37 - - [19/May/2001:15:14:29 -0500] "GET
/~ssandrew/personal.htm?UserName=Joe HTTP/1.0"
200 2438
206.223.27.37 - - [19/May/2001:15:17:12 -0500] "GET
/~ssandrew/personal.htm?UserName=Shehzaad
HTTP/1.0" 200 438
144.25.223.39 - - [19/May/2001:15:30:34 -0500] "GET /htdocs/coelist.html
HTTP/1.0" 200 4219
144.25.223.39 - - [19/May/2001:15:30:34 -0500] "GET /images/redheaderbanner.gif
HTTP/1.0" 200 1226
138.2.213.146 - - [19/May/2001:10:49:44 -0500] "GET /pls/coe/find_via_post
HTTP/1.0" 200 1119
138.2.213.146 - - [19/May/2001:10:49:44 -0500] "GET /ows-img/chalk.jpg HTTP/1.0"
404 284
130.35.35.21 - - [20/May/2001:00:36:35 -0500] "GET /images/support.jpg HTTP/1.0"
206 3106
130.35.35.21 - - [20/May/2001:00:36:35 -0500] "GET /images/ani_coe.gif HTTP/1.0"
206 73118
```

Example: Access Log with Reload Entries

The following shows an access log excerpt in which there are two Web browser reloads, followed by two shift reloads, and two more reloads:

```
138.2.213.146 - - [19/May/2001:11:04:24 -0500] "GET /cache.htm HTTP/1.0" 200 250
138.2.213.146 - - [19/May/2001:11:04:26 -0500] "GET /cache.htm HTTP/1.0" 200 250
138.2.213.146 - - [19/May/2001:11:29:24 -0500] "GET /cache.htm HTTP/1.0" 304 0
138.2.213.146 - - [19/May/2001:11:29:25 -0500] "GET /cache.htm HTTP/1.0" 304 0
138.2.213.146 - - [19/May/2001:11:29:30 -0500] "GET /cache.htm HTTP/1.0" 200 250
138.2.213.146 - - [19/May/2001:11:29:35 -0500] "GET /cache.htm HTTP/1.0" 200 250
```

Example: Access Log with Wrong Path Entry

The following 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.

```
38.2.213.146 - - [19/May/2001:10:47:30 -0500] "GET /~ssandrew/count.sh HTTP/1.0"
403 289
138.2.213.146 - - [19/May/2001:10:47:34 -0500] "GET /~ssandrew/sbin/count.sh
HTTP/1.0" 200 321
```

Example: Access Log with Status Code 404 Entry

The following 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.

```
138.2.213.146 - - [19/May/2001:10:49:44 -0500] "GET /pls/coe/find_via_post
HTTP/1.0" 200 1119
138.2.213.146 - - [19/May/2001:10:49:44 -0500] "GET /ows-img/chalk.jpg HTTP/1.0"
404 284
```

Example: Access Log with Status Code 304 Entry

The following 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 an HTTP status code of 304, indicating that document has not been modified and does not need to be returned again.

```
138.2.213.146 - - [19/May/2001:11:34:23 -0500] "GET /cache.htm HTTP/1.0" 200 250
138.2.213.146 - - [19/May/2001:11:38:23 -0500] "GET /cache.htm HTTP/1.0" 304 0
138.2.213.146 - - [19/May/2001:11:38:48 -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 Logs**.
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: Operating system-specific *Oracle HTTP Server powered by Apache Performance Guide* for TCP/IP performance tuning tips

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 > Invalidation/Statistics Port**.

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

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

The Change Invalidation/Statistics Port dialog box appears.

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

- d. From the **Protocol** list, select either **HTTP** or **HTTPS** to accept invalidation requests from one of the following URLs:

```
http://web_cache_hostname:http_port  
https://web_cache_hostname:https_port
```

Note: If you enable HTTPS for statistic monitoring requests, you cannot use the Oracle Web Cache Manager interface to send the requests. You must submit requests through the URL.

- e. 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
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 9-7 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 sent to browsers
Total Number of Bytes Saved by Compression	Additional bytes sent to browsers if compression is turned off
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	<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>
Caching Stats	<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</p> <p>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</p> <p>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: Percentage of Web browser requests for cacheable documents not served by Oracle Web Cache</p> <p>Noncacheable Misses: Percentage of Web browser requests for noncacheable documents not served by Oracle Web Cache</p>
Compression Stats	<p>This table provides information about the percentage of compressed requests that Oracle Web Cache is currently serving (% Now) and has served since it was started or restarted (% Since Start).</p> <p>Compressed Hits: Percentage of total requests served out of the cache in compressed form</p> <p>Compressed Misses: Percentage of total requests retrieved from the application Web server and compressed by Oracle Web Cache before serving</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	This table provides information about the application Web servers. It contains the following columns:
	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

10

Troubleshooting Oracle Web Cache Configuration

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

This chapter contains these topics:

- [Startup Failures](#)
- [Application Web Server Capacity](#)
- [Wrong or Older Cached Content](#)
- [Load on Oracle Web Cache Computer](#)
- [Configuration Changes Made in Oracle Web Cache Manager](#)

Startup Failures

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

- [Port Conflicts](#)
- [Cache Memory](#)
- [Privileged Ports](#)
- [Greater Than One Thousand Maximum Connections](#)
- [Wallet Cannot Be Opened](#)

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. The following shows an excerpt of `event_log` with port conflict event messages:

```
11/May/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/May/2001:11:04:42 -0800 -- Error: Failed to start the server.
11/May/2001:11:04:42 -0800 -- Error: The server could not initialize
11/May/2001:11:04:42 -0800 -- Information: The server is exiting
11/May/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 > 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, then the admin server process will not start and Oracle Web Cache Manager will not be accessible. The event log will contain messages that resemble the following output:

```
11/May/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/May/2001:10:56:11 -0800 -- Error: Failed to start the server.
11/May/2001:10:56:11 -0800 -- Error: The server could not initialize
11/May/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. The following shows an excerpt of the `webcache.xml` file with the line for the administration port shown in boldface:

```
<?xml version="1.0"?>
<!DOCTYPE CALYPSO SYSTEM "internal:///webcache.dtd">
<CALYPSO>
  <VERSION DTD_VERSION="2.0"/>
  <MULTIPOINT>
    <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"/>
  </MULTIPOINT>
```

Cache Memory

Oracle Web Cache preallocates 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 fails to start and messages that resemble the following output are written to the event log.

```
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` executable.

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`.
 6. Change the group of the executable `webcached` to the group ID you configured in Step 3.
 7. Add set-user ID permission to the `webcached` executable.
 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 listens on the privileged port with the new process identity.

See Also: Operating system documentation for information about changing ownership

Greater Than One Thousand Maximum Connections

See Also: ["Task 4: Set Resource Limits"](#) on page 5-8

As long as the executable `webcached` is owned by `root`, Oracle Web Cache can support more than 1,000 connections on UNIX. The number of connections is specified in the Current Maximum incoming connections field of the Resource Limit page (**Administering Oracle Web Cache > Resource Limits**) of Oracle Web Cache Manager.

If `webcached` is not owned by `root`, then the cache server process fails to start and messages that resemble the following output are written to the event log:

```
20/Apr/2001:18:18:24 -0800 -- Error: Could not increase number of file/socket
descriptors to 10220.
20/Apr/2001:18:18:24 -0800 -- Error: Failed to start the server.
```

To change the ownership of the `webcached` executable to `root`:

1. Stop Oracle Web Cache, and then log out of the computer.

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

2. Log back in as `root`.
3. Change the ownership of the executable `webcached` to `root`.
4. Add set-user ID permission to the `webcached` executable.
5. Log out of the computer, and re-login as the user configured in the Process Identify page (**Administering Oracle Web Cache > Process Identity**).
6. Start Oracle Web Cache.

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

See Also: Operating system documentation for information about changing ownership

Wallet Cannot Be Opened

If Oracle Web Cache is unable to open a wallet, messages that resembles the following output are written to the event log:

```
14/Sep/2001:13:16:14 -0700 -- Information: The server is exiting
14/Sep/2001:13:16:14 -0700 -- Information: OracleOraHome81WebCacheMon stopping
14/Sep/2001:13:16:24 -0800 -- Error: Oracle Wallet Failed to open at location
System Default Location. Only Auto Login Wallets Supported.
14/Sep/2001:13:16:24 -0800 -- Error: The server could not initialize
14/Sep/2001:13:16:24 -0800 -- Information: The server is exiting
14/Sep/2001:13:16:24 -0800 -- Error: Cannot open log files because NULL socket
indicates problem
```

To resolve this error, perform the procedure that follows. At the end of each step, restart Oracle Web Cache with the `webcachectl start` command, and recheck the `event_log` file for the "Oracle Wallet Failed..." error.

1. Follow the ["Enabling Wallets to Open on Windows"](#) on page 5-19 to ensure that the wallet can be opened at startup.
2. Ensure that the wallet directory exists:
 - `/etc/ORACLE/WALLETS/user_name` on UNIX
 - `%USERPROFILE%\ORACLE\WALLETS` on Windows NT and Windows 2000
3. Ensure that wallet files `cwallet.sso` and `ewallet.der` exist.

If these files do not exist, then the wallet does not exist.

If these files do exist, then the wallet user may not match Oracle Web Cache user. Continue to Step 4.

4. Locate the <WALLET> line in the webcache.xml file:

```
<?xml version="1.0"?>
<!DOCTYPE CALYPSO SYSTEM "internal:///webcache.dtd">
<CALYPSO>
  <VERSION DTD_VERSION="2.0"/>
  <MULTIPOINT>
    <LISTEN IPADDR="ANY" PORT="1100" PORTTYPE="NORM"/>
    <LISTEN IPADDR="ANY" PORT="1101" PORTTYPE="NORM"
SSLENABLED="SSLV3_V2H"/>
    <LISTEN IPADDR="ANY" PORT="4000" PORTTYPE="ADMINISTRATION"/>
    <LISTEN IPADDR="ANY" PORT="4001" PORTTYPE="INVALIDATION"/>
    <LISTEN IPADDR="ANY" PORT="4002" PORTTYPE="STATISTICS"/>
  </MULTIPOINT>
  <WALLET>file:C:\oracle\ora81\webcache\wallets\client</WALLET>
```

Add a PASSWORD attribute to the <WALLET> element as follows:

```
<WALLET PASSWORD="password">wallet_location</WALLET>
```

The following example shows a wallet password of oracle:

```
<WALLET
PASSWORD="oracle">file:C:\oracle\ora81\webcache\wallets\client</WALLET>
```

5. Confirm that the following conditions do not apply:

- The user that saved the wallet does not match the user that starts the webcached and webcachectl executables.

Change the user name of wallet to match the user that starts the webcached and webcachectl executables.

- The wallet was copied from one computer to another.

Create the wallet on the computer with the Oracle Wallet Manager.

See Also:

["Task 3: Specify Web Site Settings"](#) on page 5-5 for instructions on setting the user ID of the Oracle Web Cache executables

["Creating a Wallet"](#) on page 5-17 for instructions on creating the wallet

6. Ensure that Oracle Wallet Manager can open the wallet.

If Oracle Wallet Manager cannot open the wallet, then the wallet is corrupt. In this case, recreate the wallet.

See Also: ["Creating a Wallet"](#) on page 5-17

Application Web Server Capacity

If an application Web server has reached capacity, then 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: Specify Web Site Settings"](#) 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, then error messages about the cacheability rules will be reported.

See Also: ["Configuring Event Logs"](#) on page 8-25 for further information about turning on verbose logging

Load on Oracle Web Cache Computer

On UNIX operating systems, the `top` and `uptime` utilities report a higher than expected average load when the Oracle Web Cache computer is idle. This occurs because Oracle Web Cache performs light maintenance work, even when it is idle. During idle mode, the following effect occurs:

- The uptime load—the average kernel scheduler queue length—is going to be longer. Oracle Web Cache increases the average queue length (uptime output) by roughly 1.
- The CPU load is still low because the work Oracle Web Cache performs is minimal.

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.

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.

See Also:

- ["Starting and Stopping Oracle Web Cache"](#) on page 8-2
- ["Status Messages"](#) on page 4-4 for further information about status message in Oracle Web Cache Manager

Note: When you stop Oracle Web Cache, all objects are cleared from the cache. In addition, all statistics are cleared.

A Case Study Deployment

This chapter provides a case study of how Digital River, a global Commerce Service Provider (CSP), deployed Oracle Web Cache, release 1.0.2.

This case study describes the challenges that Digital River faced and the solutions that Oracle Web Cache provided. It describes Oracle Web Cache implementation details, including deployment, cacheability rules, invalidation mechanisms, and overall performance.

This chapter contains these topics:

- [About Digital River](#)
- [Why Oracle Web Cache?](#)

About Digital River

Founded in 1994, Digital River is a leading global CSP, offering complete e-commerce systems and services to over 8,000 clients. Digital River's commerce services include e-commerce strategy, site development and hosting, order and transaction management, systems integration, product fulfillment and returns, e-marketing, and customer service. Some of Digital River's clients include Symantec, Autodesk, Major League Baseball, MLB Allstar voting, 3M, Polairs, Nabisco Gifts, and H & R Block.

Content of Digital River

Much of the content delivered by Digital River is dynamically generated. This content consists of pages that support session cookies for identifying users. Once users are identified, they are assigned a promotional campaign that is generated for each page delivered.

Hardware/Network Deployment of Digital River Before Oracle Web Cache

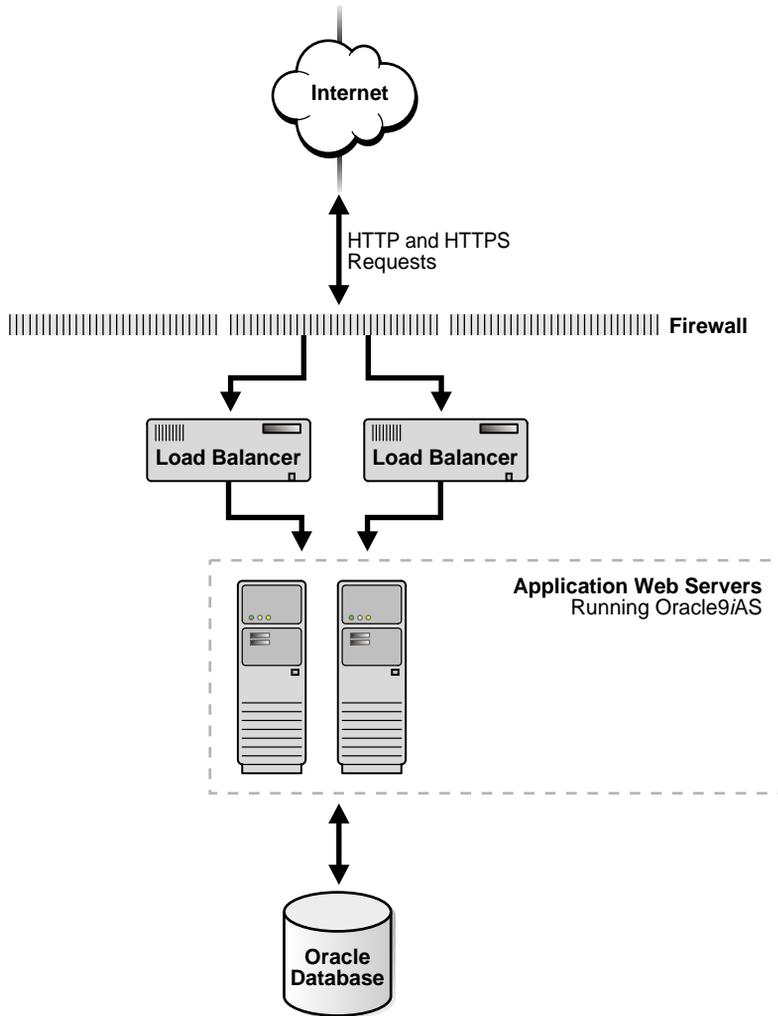
Requests were serviced by one of two load balancers running in active-active mode. Two load balancers managed and distributed incoming HTTP and HTTPS requests equally among the application Web servers.

The middle-tier featured two application Web servers that ran Oracle9i Application Server (Oracle9iAS) release 1.0. To ensure high availability, each application Web server was configured with identical content. They are Sun Enterprise 6500 servers, each with 10 GB of RAM and 10 x 336 MHz processors

On the back end, Digital River deployed a primary Oracle8i release 8.1.6 database, as well as standby databases. The primary database ran on a Sun Enterprise 6500 server with 20 GB of RAM and 20 x 336 MHz processors. Pages were generated dynamically with PL/SQL stored procedures, which were invoked with the `mod_plsql` module on the application Web servers.

The overall deployment is depicted in [Figure 11-1](#).

Figure 11-1 *Digital River Deployment Before Oracle Web Cache*



Why Oracle Web Cache?

Digital River currently generates more than 1.5 million page views for each day, a figure that doubles in the last quarter of the year with the holiday shopping rush. While the Digital River deployment provided high availability, it placed the burden of incoming requests on the application Web servers and the backend databases. In fact, due primarily to the dynamic generation of pages, the backend database was sustaining 300 concurrent connections at any given time.

To sustain ever-mounting traffic levels and to support a growing client base, Digital River wanted a more scalable, high-performance application Web server architecture. Digital River considered replacing the Sun Enterprise 6500 servers with more expensive Sun Enterprise 10000 servers, but decided the cost outweighed the benefits. In addition, Digital River wanted to improve overall performance by reducing the burden on the backend databases. In the end, Digital River chose a more scalable application Web server architecture and decided to deploy Oracle Web Cache to reduce the load on the backend databases.

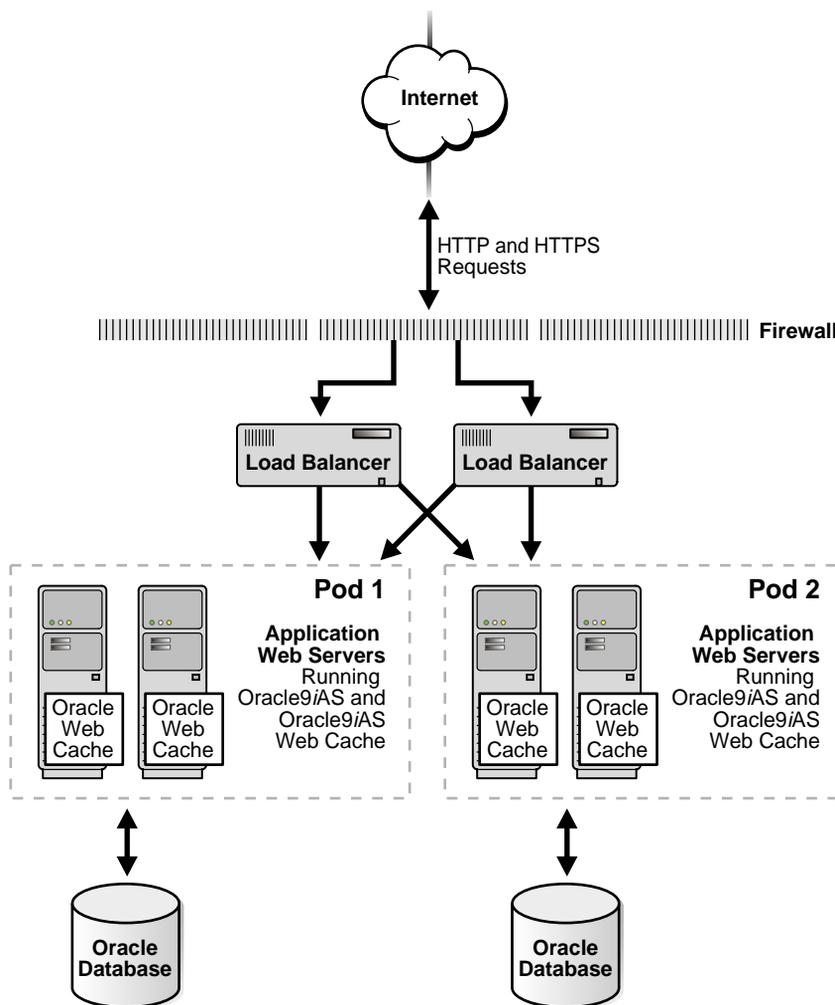
Hardware/Network Deployment with Oracle Web Cache

To take advantage of Oracle Web Cache's caching features, Digital River migrated from Oracle9iAS release 1.0 to Oracle9iAS release 1.0.2.

To provide a more scalable solution that enables Digital River to incrementally add less expensive servers, Digital River created two pods of two application Web servers. Each of the application Web servers in the pod was configured with Oracle9iAS and Oracle Web Cache. Low-cost Sun Enterprise 420R computers, each with 4 GB of RAM and 4 x 450 MHz processors, replaced the Sun Enterprise 6500 servers. Each pod was configured with a dedicated Oracle8i release 8.1.6 database.

The new deployment is depicted in [Figure 11-2](#).

Figure 11-2 Digital River Deployment with Oracle Web Cache



Cache Configuration

The main content that Digital River wanted to cache was its dynamic content. Prior to Oracle Web Cache, dynamic content was generated from the backend database. When a user first accessed a Web site hosted by Digital River, the request was sent to the backend database. In turn, the database set a session cookie associated with a list of promotional campaigns. The database sent the session cookie back to the client and served the dynamically generated page.

In order to make this content cacheable, Digital River changed its dynamic generation so that the database sends the browser to a redirected URL that supports the appropriate campaign ID for that user. This redirected URL is then cached by Oracle Web Cache. By making this minimum application change, Digital River is able to achieve greater than 80 percent cache hit rates.

The Digital River team established the same cacheability rules for each Oracle Web Cache deployment with Oracle Web Cache Manager. The cacheability rules are shown in [Figure 11-3](#).

Figure 11-3 Cacheability Rules

The screenshot shows the Oracle Web Cache Manager administration interface. The main content area displays the 'Cacheability Rules' table. The table has the following data:

Select	Priority	URL Expression	Cache/Don't Cache	GET/POST Caching	Detailed Settings
<input type="radio"/>	1	.*CACHE_ID=[0-9]	Cache	GET, GET with query string	details
<input type="radio"/>	2	\.jpg\$	Cache	GET, GET with query string	details
<input type="radio"/>	3	\.pdf\$	Don't Cache	N/A	N/A
<input type="radio"/>	4	\.html\$	Cache	GET, GET with query string	details

Below the table are control buttons: 'Edit Selected...', 'Delete Selected...', 'Insert Above...', 'Insert Below...', 'Move Up', and 'Move Down'.

Table 11–1 describes the cacheability rules.

Table 11–1 Digital River Cacheability Rules

Priority Order	URL Expression	Cache/Don't Cache	Description
1	.*CACHE_ID=[0-9]	Cache	Caches all the redirected URL pages
2	\.jpg\$	Cache	Caches pages ending in .jpg, and expires them after two hours in the cache.
3	\.pdf\$	Don't Cache	Does not cache pages ending in .pdf
4	\.html\$	Cache	Caches all HTML pages

Digital River provides a content-management system for its customers to use. This system runs automated invalidation scripts, ensuring that content changes are reflected in the cache.

Performance Results with Oracle Web Cache

The overall cache hit rate on Oracle Web Cache is 86 percent, reducing the number of concurrent connections to the backend database from 300 to 30. In addition to the 10-fold load reduction on the backend databases, pages are being delivered up to 10 times faster since deploying Oracle Web Cache. Prior to Oracle Web Cache, page response time was at least 3 seconds. With Oracle Web Cache, delivery is now between 0.1 and 0.3 seconds.

Part III

Reference

Part III provides reference material for this guide.

This part contains these appendixes:

- [Appendix A, "Oracle Web Cache Directory Structure"](#)
- [Appendix B, "Oracle Web Cache Default Settings"](#)
- [Appendix C, "Invalidation Document Type Definition"](#)
- [Appendix D, "Edge Side Includes Language"](#)
- [Appendix E, "Event Log Messages"](#)

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 `$ORACLE_HOME/webcache` directory on UNIX and an `ORACLE_HOME\webcache` directory on Windows. [Table A-1](#) describes the directory structure components of the `$ORACLE_HOME/webcache` directory on UNIX and the `ORACLE_HOME\webcache` directory on Windows.

Table A-1 Oracle Web Cache Directory Structure

Directory/File	Contents
/bin directory on UNIX \bin in the <code>ORACLE_HOME</code> directory on Windows	Contains the Oracle Web Cache binaries, including the webcached main executable and the webcachectl command-line tool. On UNIX, webcachectl is symbolically linked to the <code>\$ORACLE_HOME/bin</code> directory, which is set in the path during installation.
/docs directory on UNIX \docs directory on Windows	Contains documentation and online help for Oracle Web Cache Manager
/examples directory on UNIX \examples directory on Windows	Contains unsupported Oracle Web Cache scripts for invalidation and personalized attributes and system administration scripts for debugging. Read <code>readme.examples.html</code> in the directory for further information about the scripts.
<code>internal.xml</code> and <code>internal_admin.xml</code> files	Contains internal configuration settings
/lib directory on UNIX	Contains library files
/invalidation directory on UNIX \invalidation directory on Windows	Contains the Document Type Definition (DTD) file <code>WCSinvalidation.dtd</code> for invalidation requests and responses

Directory/File	Contents
/logs directory on UNIX \logs directory on Windows	Contains event and access logs
/mesg directory on UNIX \mesg directory on Windows	Contains message files
readme.html	Contains important release information
/utl directory on UNIX \utl directory on Windows	Contains a utility to convert release 1.0 configuration files to a format supported by release 2.0
webcache.xml	Contains configuration parameters set by Oracle Web Cache Manager

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	Location in Oracle Web Cache Manager to 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	Location in Oracle Web Cache Manager to 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 ORACLE_HOME\webcache\logs on Windows	This file name and default directory cannot be modified.
Access logs	access_log in \$ORACLE_HOME/webcache/logs on UNIX and ORACLE_HOME\webcache\logs on Windows	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	900	Administering Oracle Web Cache > Resource Limits

Invalidation Document Type Definition

This appendix describes the Document Type Definition (DTD), or grammar, of invalidation requests and responses. The DTD for requests and responses is defined within `wcsinvalidation.dtd` in the `$ORACLE_HOME/webcache/invalidation` directory on UNIX, and the `ORACLE_HOME\webcache\invalidation` directory on Windows.

This appendix contains these topics:

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

Invalidation Request DTD

Figure C-1 shows the DTD for invalidation messages.

Figure C-1 Invalidation Request DTD

```

<!-- root element for invalidation request -->
<!ELEMENT   INVALIDATION      (SYSTEM?,OBJECT+)>

<!-- VERSION is currently "WCS-1.0" without the quotes -->
<!ATTLIST  INVALIDATION
           VERSION            CDATA            #REQUIRED
>

<!ELEMENT  OBJECT              ((BASICSELECTOR|ADVANCEDSELECTOR), ACTION)>

<!ELEMENT  BASICSELECTOR      EMPTY>
<!ATTLIST  BASICSELECTOR
           URI                 CDATA            #REQUIRED
>

<!ELEMENT  ADVANCEDSELECTOR   (COOKIE|HEADER|OTHER)*>
<!ATTLIST  ADVANCEDSELECTOR
           URIPREFIX          CDATA            #REQUIRED
           URIEXP             CDATA            #IMPLIED
           METHOD              CDATA            #IMPLIED
           BODYEXP            CDATA            #IMPLIED
>

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

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

<!ELEMENT  ACTION             EMPTY>
<!ATTLIST  ACTION
           REMOVALTTL         CDATA            #IMPLIED
>

```

Table C-1 Invalidation Request DTD Elements and Attributes

Invalidation Element	Invalidation Attribute	Description
INVALIDATION		
	VERSION	Version of the DTD
OBJECT		
	BASICSELECTOR	Invalidation based on the URL
	ADVANCEDSELECTOR	Invalidation based on advanced invalidation selectors
	ACTION	Action to perform on document(s)
BASICSELECTOR		
	URI	URL of the document(s) to be invalidated
ADVANCEDSELECTOR		
	URIPREFIX	Prefix path (beginning and ending with "/") of the document(s) to be invalidated. The prefix is interpreted literally, including reserved regular expression characters.
	URIEXP	URL of the document(s) to be invalidated underneath the URIPREFIX
	METHOD	HTTP request method of the document(s) to be invalidated
	BODYEXP	HTTP POST body message of the document(s) to be invalidated
COOKIE		
	NAME	Cookie name used by the documents contained within the URL
	VALUE	Value of the cookie If no value is present, then only documents with the named cookie but without value are invalidated.

Invalidation Element	Invalidation Attribute	Description
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.
ACTION	REMOVALTTL	Maximum time that documents can reside in the cache before they are invalidated. The default is 0 seconds.

Invalidation Response DTD

Figure C-2 shows the DTD for invalidation responses.

Figure C-2 Invalidation Response DTD

```

<!-- root element for invalidation result -->
<!ELEMENT   INVALIDATIONRESULT (SYSTEM?, OBJECTRESULT+)>

<!-- VERSION is currently "WCS-1.0" without the quotes -->
<!ATTLIST   INVALIDATIONRESULT
            VERSION          CDATA          #REQUIRED
>

<!ELEMENT   OBJECTRESULT      ((BASICSELECTOR|ADVANCEDSELECTOR), RESULT)>

<!ELEMENT   RESULT            EMPTY>
<!ATTLIST   RESULT
            ID                CDATA          #REQUIRED
            STATUS            CDATA          #REQUIRED
            NUMINV            CDATA          #REQUIRED
>

```

Table C-2 Invalidation Response DTD Elements and Attributes

Invalidation Element	Invalidation Attribute	Description
RESULT	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 continues 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 that are not cacheable ■ URI NOT FOUND for documents not found
	NUMINV	Number of documents invalidated

Edge Side Includes Language

This appendix describes the **Edge Side Includes (ESI)** tag library provided for content assembly of dynamic HTML fragments.

This appendix contains these topics:

- [Overview of ESI Tag Library](#)
- [ESI Tag Descriptions](#)

Overview of ESI Tag Library

ESI is an open specification co-authored by Oracle Corporation, the purpose being to develop a uniform programming model to assemble dynamic pages on the edge of the Internet.

ESI is an XML-based markup language that enables dynamic content assembly of fragments by Oracle Web Cache. A template page is configured with ESI markup tags that fetch and include dynamic HTML fragments. The fragments themselves can also contain ESI markup. You can assign cacheability rules to the template page and HTML fragments. By enabling Oracle Web Cache to assemble dynamic pages rather than the application Web server, you can increase the overall cacheable content.

See Also:

- ["Configuring Pages for Content Assembly and Partial Page Caching"](#) on page 6-33
- ["Configuring Cacheability Attributes in Response Headers"](#) on page 6-45
- <http://www.edge-delivery.org> for additional information about the ESI language

The following topics provide an overview of ESI usage:

- [Syntax Rules](#)
- [Nesting Elements](#)
- [Variable Expressions](#)
- [Exceptions and Errors](#)
- [Syntax Rules](#)

Syntax Rules

ESI elements and attributes adhere to XML syntax but can be embedded in other documents such as HTML or XML documents. When Oracle Web Cache processes the page, the ESI elements themselves are stripped from the output.

ESI syntax generally adheres to XML syntax rules. Keep the following in mind when using the tags:

- ESI tags and attributes must be lowercase
- Supported CGI environment variables require uppercase
- ESI does not support the use of whitespace next to the equal sign (=) or between the "<" and "esi:"

The following shows an invalid construction:

```
<esi:include src = "www.foo.com"/>
```

The following shows the correct form:

```
<esi:include src="www.foo.com"/>
```

Nesting Elements

As shown in [Figure D-1](#), an ESI tag can contain nested ESI elements and other HTML markup.

Figure D-1 *Nested ESI Elements*

```
<esi:choose>
  <esi:when test="\$(HTTP_HOST) == 'www.company.com' ">
    <esi:include src="/company.html" />
    <h4>Another</h4>
    <esi:include src="/another.html" />
  </esi:when>
  <esi:when test="\$(HTTP_COOKIE{fragment}) == 'First Fragment' ">
    <esi:try>
      <esi:attempt>
        <esi:include src="/fragment1.html" />
      </esi:attempt>
      <esi:except>
        <esi:choose>
          <esi:when test="\$(HTTP_COOKIE{otherchoice}) == 'image' ">
            
          </esi:when>
        </esi:choose>
      </esi:except>
    </esi:try>
  </esi:when>
</esi:choose>
```

```

        <esi:otherwise>
            The fragment is unavailable.
        </esi:otherwise>
    </esi:choose>
</esi:except>
</esi:try>
</esi:when>
<esi:otherwise>
    The default selection.
</esi:otherwise>
</esi:choose>

```

Variable Expressions

Table D–1 lists the variables that are supported by ESI. Except for `QUERY_STRING`, the values for the variables are taken from HTTP request-header fields. In the case of `QUERY_STRING`, the value is taken from either the HTTP POST body or the URL. Variables are only interpreted when enclosed within ESI tags.

Table D–1 ESI-Supported Variables

Variable Name	HTTP Request-Header Field	Substructure Type	Description	Example
<code>HTTP_ACCEPT_LANGUAGE</code>	Accept-Language	List	Specifies in a comma-separated list the set of languages that are preferred as a response	<code>da, en-gb, en</code>
<code>HTTP_COOKIE</code>	Cookie	Dictionary	Specifies in a semi-colon-separated list the cookie name and value pairs	<code>id=571; visits=42</code>
<code>HTTP_HOST</code>	Host	Not Applicable	Specifies the host name and port number of the resource. Port 80 is the default port number.	<code>http://www.company.com</code>
<code>HTTP_REFERER</code>	Referer	Not Applicable	Specifies the URL of the reference resource	<code>http://www.company.com</code>
<code>HTTP_USER_AGENT</code>	User-Agent	Dictionary	Specifies the Web browser type, browser version, or operating system that initiated the request.	<code>Mozilla, MSIE 5.5</code>
<code>QUERY_STRING</code>	Not Applicable	Dictionary	Ampersand-separated list of string and value pairs	<code>first=Jane&last=Doe</code>

Usage

Variable names must be in uppercase.

To reference a variable, surround the variable name with parenthesis and append a dollar sign:

```
$( VARIABLE_NAME )
```

For example:

```
$( HTTP_HOST )
```

Variable Substructure Access

Variables with a substructure type of List or Dictionary in [Table D-1](#) are accessed by a key as follows:

```
$( VARIABLE_NAME {key} )
```

To access a variable's substructure, append the variable name with braces containing the key which is being accessed. For example:

```
$( HTTP_COOKIE {username} )
```

The key is case sensitive.

Variables identified with a substructure type of Dictionary in [Table D-1](#) make access to strings available through their appropriate keys. Dictionary keys are case sensitive.

Variables identified with a substructure type of List in [Table D-1](#) return a boolean value depending on whether the requested value is present.

[Table D-2](#) lists examples of substructure access, with specific values based on the examples in [Table D-1](#) on page D-4.

Note: `HTTP_USER_AGENT` can use one of three keys: `browser` for browser type, `version` for browser version, and `os` for operating system.

Table D-2 Dictionary and List Examples

Variable Name and Key	Example Variable Setting	Returned Value
<code>\$(HTTP_ACCEPT_LANGUAGE{language})</code>	<code>\$(HTTP_LANGUAGE{en-gb})</code>	If the <code>Accept-Language</code> header uses <code>en-gb</code> , then returns a value of <code>true</code> .
<code>\$(HTTP_COOKIE{cookie_name})</code>	<code>\$(HTTP_COOKIE{visits})</code>	If the <code>Cookie</code> header contains the string <code>visits=42</code> , then returns a value of <code>42</code> .
<code>\$(HTTP_USER_AGENT{browser,version,os})</code>	<code>\$(HTTP_USER_AGENT{browser})</code>	Enables Oracle Web Cache to determine the User-Agent's browser. Returns values of <code>Mozilla</code> , <code>MSIE</code> , or <code>Other</code> .
<code>\$(HTTP_USER_AGENT{browser,version,os})</code>	<code>\$(HTTP_USER_AGENT{version})</code>	Enables Oracle Web Cache to determine the User-Agent's version. Returns the version number of the browser.
<code>\$(HTTP_USER_AGENT{browser,version,os})</code>	<code>\$(HTTP_USER_AGENT{os})</code>	Enables Oracle Web Cache to determine the User-Agent's operating system. Returns a value of <code>WIN</code> , <code>MAC</code> , <code>UNIX</code> , or <code>Other</code> .
<code>\$(QUERY_STRING{string})</code>	<code>\$(QUERY_STRING{last})</code>	Returns a value of <code>Doe</code>

Variable Default Values

Variables with empty values or nonexistent values, or variables with undefined keys evaluate to an empty string when they are accessed. You can use the logical or (`|`) operator to specify a default value in the following form:

```
$(VARIABLE|default)
```

The following example results in Oracle Web Cache fetching

`http://example.com/default.html` if the cookie `id` is not in the request:

```
<esi:include src="http://example.com/$(HTTP_COOKIE{id}|default).html"/>
```

As with other literals, if whitespace needs to be specified, then the default value must be single-quoted. For example:

```
$(HTTP_COOKIE{first_name} | 'new user')
```

Note: HTTP_HOST and HTTP_REFERER do not support default values in this release.

Exceptions and Errors

ESI uses two mechanisms for exception and error handling. In a given situation, you can make use of both mechanisms simultaneously, use one at a time, or use neither, depending on the business logic you are developing. The mechanisms are described in the following topics:

- [ESI Language Control](#)
- [Apology Page](#)

Apology Page

The first mechanism is to use an apology page.

Set the apology page in the **Administering Web Sites > Apology Page** of Oracle Web Cache Manager.

ESI Language Control

The second mechanism is found in the ESI language, which provides two specific elements for fine-grain control over content assembly in error scenarios:

The `onerror` attribute of the `<esi:include>` tag

The `try | attempt | except block`

See Also:

- ["ESI include Tag"](#) on page D-9
- ["ESI try | attempt | except Tags"](#) on page D-14

Enabling ESI

To enable Oracle Web Cache to process ESI tags, an HTTP `Surrogate-Control` header is set in the HTTP response message of the pages that use ESI tags.

ESI Tag Descriptions

This section describes the following ESI tags, which are used for partial page caching operations:

- [ESI include Tag](#)
- [ESI choose | when | otherwise Tags](#)
- [ESI try | attempt | except Tags](#)
- [ESI comment Tag](#)
- [ESI remove Tag](#)
- [ESI <!--esi-->Tag](#)
- [ESI vars Tag](#)

ESI include Tag

The `<esi:include>` tag provides syntax for including fragments. It specifies the object to include and allows for two additional, optional settings.

Syntax

```
<esi:include src="URL_fragment" onerror="continue"/>
```

Note that `<esi:include>` does not have a closing `</esi:include>`.

Attributes

- `src`—Specifies the HTML fragment to fetch. The fragment can be a file referenced by a URL or it can include variables.
- `onerror`—**Optional**. Specifies that if the fetch failed on the `src` object, to ignore the ESI tag and serve the page.

Note: The ESI language release 1.0 specification provides support for an `alt` attribute, which specifies an alternative resource if the `src` is not found. Because Oracle Web Cache is near the application Web server, the `alt` tag cannot be processed in a useful manner. Therefore, Oracle Web Cache ignores the `alt` attribute.

Syntax Usage

- `<esi:include>` supports up to three levels of recursion.
- `<esi:include>` does not support escaped double quotes (`\`). For example, the following is not supported:

```
<esi:include src="file\"user.htm"/>
```

- `src` and `onerror` attributes do not need to be in a particular order

Usage

The `<esi:include>` tag tells Oracle Web Cache to fetch the fragment specified by the `src` attribute. The attribute value must be a valid URL. Relative URLs will be resolved relative to the template page. The resulting object will replace the element in the markup served to the browser. The included fragment must reside on the same site. Therefore, it is not necessary to specify the host name in the URL.

If the include is successful, the contents of the fetched `src` URL display. The included object is included exactly at the point of the include tag. For example, if the include tag is in a table cell, the fetched object is displayed in the table cell.

If Oracle Web Cache cannot fetch the `src`, it returns an HTTP status code greater than 400 with an error message, unless the `onerror` attribute is present. If `onerror="continue"` is specified, then Oracle Web Cache ignores the `<esi:include>` tag.

Examples

The following ESI markup includes a file named `frag1.htm`:

```
<esi:include src="/frag1.htm" onerror="continue"/>
```

The following ESI output includes the result of a dynamic query:

```
<esi:include src="/search?query=${QUERY_STRING(query)"/>
```

ESI choose | when | otherwise Tags

The `<esi:choose>`, `<esi:when>`, and `<esi:otherwise>` conditional tags provide the ability to perform logic based on boolean expressions.

Syntax

```
<esi:choose>
  <esi:when test=BOOLEAN_expression>
    Perform this action
  </esi:when>
  <esi:when test="BOOLEAN_expression">
    Perform this action
  </esi:when>
  <esi:otherwise>
    Perform this other action
  </esi:otherwise>
</esi:choose>
```

Attributes

`test`—Specifies the boolean operation

Usage

- Each `<esi:choose>` tag must have a least one `<esi:when>` tag, and may optionally contain exactly one `<esi:otherwise>` tag.
- Oracle Web Cache will execute the first `<esi:when>` tag whose test attribute evaluates truthfully, and then exit the `<esi:choose>` tag. If no `<esi:when>` tag evaluates to true and an `<esi:otherwise>` tag is present, then that element's content will be executed.
- Other HTML or ESI element can be included inside `<esi:when>` or `<esi:otherwise>` elements.

Boolean Expressions

The `test` attribute uses boolean expressions to determine how to evaluate true or false logic. ESI supports the following boolean operators:

- `==` (equal to)
- `!=` (not equal to)
- `>` (greater than)
- `<` (less than)
- `>=` (greater than or equal to)
- `<=` (less than or equal to)
- `&` (and)
- `|` (or)
- `!` (not)

Note the following about the use of boolean expressions:

- Operands associate from left to right.
Sub-expressions can be grouped with parentheses in order to explicitly specify association.
- If both operands are numeric, then the expression is evaluated numerically.
- If either operand is non-numeric, then both operands are evaluated as strings.
For example, `'a'==3` evaluates to `'a'=='3'`, where 3 is evaluated as a string.
- The comparison of two boolean expressions results in an undefined operation.
- If an operand is empty or undefined, then the expression always evaluates to false.

- The logical operators (&, !, and |) are used to qualify expressions, but cannot be used to make comparisons.

- Use single quotes (') for constant strings.

For example, the following string is a valid construction:

```
$(HTTP_COOKIE{name})=='typical'
```

- Escaped single quotes (\') are not permitted. For example, the following is not supported:

```
$(HTTP_COOKIE{'user\'s name'})=='typical'
```

- Arithmetic operations and assignments are not permitted.

The following expressions show correct usage:

```
!(1==1)
>('a'<='c')
(1==1)|('abc'=='def')
(4!=5)&(4==5)
```

The following expressions show incorrect usage:

```
(1 & 4)
("abc" | "edf")
```

Statements

Statements must be placed inside an `<esi:when>` or `<esi:otherwise>` subtag. Statements outside the subtags cannot be evaluated as conditions. [Figure D-2](#) shows invalid placement of statements.

Figure D-2 Statement Placement

```
<esi:choose>
  HTML text. This is invalid because any characters other than whitespace
  are not allowed in this area.
  <esi:when test="$(HTTP_HOST) == 'www.company.com'">
    <esi:include src="/company.html" />
  </esi:when>
  HTML text. This is invalid because any characters other than whitespace
  are not allowed in this area.
  <esi:when test="$(HTTP_COOKIE{fragment}) == 'First Fragment'">
    
  </esi:when>
  HTML text. This is invalid because any characters other than whitespace
  are not allowed in this area.
  <esi:otherwise>
```

The default selection.

```
</esi:otherwise>
```

HTML text. This is invalid because any characters other than whitespace are not allowed in this area.

```
</esi:choose>
```

Example

The following ESI markup includes `advanced.html` for requests that use the cookie `Advanced` and `basic.html` for requests that use the cookie `Basic`:

```
<esi:choose>
  <esi:when test="\$(HTTP_COOKIE{group})=='Advanced'">
    <esi:include src="http://www.company.com/advanced.html"/>
  </esi:when>
  <esi:when test="\$(HTTP_COOKIE{group})=='Basic User'">
    <esi:include src="http://www.company.com/basic.html"/>
  </esi:when>
  <esi:otherwise>
    <esi:include src="http://www.company.com/new_user.html"/>
  </esi:otherwise>
</esi:choose>
```

ESI try | attempt | except Tags

The `<esi:try>` tag provides for exception handling. `<esi:try>` must contain exactly one instance of both an `<esi:attempt>` and an `<esi:except>` tag:

Syntax

```
<esi:try>
  <esi:attempt>
    Try this...
  </esi:attempt>
  <esi:except>
    If the attempt fails, then perform this action...
  </esi:except>
</esi:try>
```

Usage

Oracle Web Cache first processes the contents of `<esi:attempt>`.

A failed `<esi:attempt>` triggers an error and causes Oracle Web Cache to process the contents of the `<esi:except>` tag.

Example

The following ESI markup attempts to fetch an ad. If the ad cannot be included, Oracle Web Cache includes a static link instead.

```
<esi:try>
  <esi:attempt>
    <esi:comment text="Include an ad"/>
    <esi:include src="http://www.company.com/ad1.htm"/>
  </esi:attempt>
  <esi:except>
    <esi:comment text="Just write some HTML instead"/>
    <a href=www.company.com>www.company.com</a>
  </esi:except>
</esi:try>
```

ESI comment Tag

The `<esi:comment>` tag enables you to comment ESI instructions, without making the comments available in the processor's output.

Syntax

```
<esi:comment text="text commentary"/>
```

`<esi:comment>` is an empty element, and does not have an end tag.

Usage

The `<esi:comment>` tag is not evaluated by Oracle Web Cache. If comments need to be visible in the HTML output, use standard XML/HTML comment tags.

Example

The following ESI markup provides a comment for an included GIF file:

```
<esi:comment text="the following animation will have a 24 hour TTL"/>  
<esi:include src="http://www.company.com/logo.gif" onerror="continue" />
```

ESI remove Tag

The `<esi:remove>` tag allows for specification of non-ESI markup output if ESI processing is not enabled with the `Surrogate-Control` header or there is not an ESI-enabled cache.

Syntax

```
<esi:remove>...HTML output</esi:remove>
```

Usage

Any HTML or ESI elements can be included within this tag, except other `<esi:remove>` tags. Note that nested ESI tags are not processed.

Example

The following ESI markup includes `http://www.company.com` if the `<esi:include>` content cannot be included.

```
<esi:include src="http://www.company.com/ad.html"/>
<esi:remove>
  <a href="http://www.company.com">www.company.com</a>
</esi:remove>
```

Normally, when Oracle Web Cache processes this example block, it fetches the `ad.html` file and includes it into the template page while silently discarding the `<esi:remove>` tag and its contents. If ESI processing is not enabled, all of the elements are passed through to browser, which ignores ESI markup. However, the browser displays the ` HTML` link.

ESI <!--esi-->Tag

The <!--esi...--> tag enables HTML marked up with ESI tags to display to the browser without processing the ESI tags. When a page is processed with this tag, Oracle Web Cache removes the starting <!--esi and ending end --> elements, while still processing the contents of the page. When the markup cannot be processed, this tag assures that the ESI markup will not interfere with the final HTML output.

Syntax

```
<!--esi
  ESI elements
-->
```

Usage

Any ESI or HTML elements can be included within this tag, except other <!--esi...--> tags.

Example

The following ESI markup hides the "Hello, *Name*" greeting if the ESI markup cannot be processed.

```
<!--esi
  <p><esi:vars>Hello, ${HTTP_COOKIE{name}}!</esi:vars></p>
-->
```

If the ESI markup can be processed, then <!--esi and --> are removed in the final output. The output displays only <p><esi:vars>Hello, \${HTTP_COOKIE{name}}!</esi:vars></p>.

ESI vars Tag

The `<esi:vars>` tag enables you to use an ESI environment variable outside of an ESI tag.

Syntax

```
<esi:vars>ESI Variable</esi:vars>
```

Usage

See Also: ["Variable Expressions"](#) on page D-4

Example

The following ESI markup includes the `cookie type` and its value as part of the included URL:

```
<esi:vars>
  
</esi:vars>
```

The following ESI markup includes the user's `sessionID` and `category type` cookie values as part of the `` links.

```
<esi:vars>
  <a href="/shopping.jsp?sessionID=${QUERY_STRING{sessionID}}&type=${QUERY_
STRING{type}}">
  
  </a>
  <a href="/news.jsp?sessionID=${QUERY_STRING{sessionID}}&type=${QUERY_
STRING{type}}">
  
  </a>
  <a href="/sports.jsp?sessionID=${QUERY_STRING{sessionID}}&type=${QUERY_
STRING{type}}">
  
  </a>
  <a href="/fun.jsp?sessionID=${QUERY_STRING{sessionID}}&type=${QUERY_
STRING{type}}">
  
  </a>
  <a href="/about.jsp?sessionID=${QUERY_STRING{sessionID}}&type=${QUERY_
STRING{type}}">
  
  </a>
</esi:vars>
```

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 E-1](#) lists the common event log informational messages.

Table E-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 port <i>port</i> address <i>ip_address</i>	The listening port number and IP address for Web browser requests to Oracle Web Cache.
Listening on STATISTICS port <i>port</i> 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 admin server process started the cache server process.
The admin server started successfully	The Oracle Web Cache admin server process successfully started.
The cache server started successfully	The Oracle Web Cache cache server 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 is 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. URI is not invalidated	The URL is not in the cache and cannot be invalidated.

Warning Events

Table E-2 lists the common event log warning messages.

Table E-2 Warning Events

Message	Description
Startup Initialization Events	
The admin server couldn't start the cache server, running in admin-only mode	The admin server process is unable to start the cache server process. This may due to a listening port conflict.
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.

Message	Description
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.
Origin Server module got an ABORT; Errno: <i>error_number</i> URI: <i>URI</i>	<p>Oracle Web Cache detects a problem with the application Web server. <i>error_number</i> can be one of the following:</p> <ul style="list-style-type: none"> 3 - low memory There is not enough memory for the webcached executable to run. 4 - LB fail All of the application Web servers are down, disabling the load balancing feature. 5 - socket connect fail The application Web server is down. 6 - socket create fail The Oracle Web Cache computer has run short of system resources. 7 - send request fail The request to the application Web server has failed. 8 - bad input stream to send Oracle Web Cache is unable to send the HTTP request body to the application Web server. 9 - recv fail Oracle Web Cache is unable to receive responses from the application Web server. 10 - create header fail Oracle Web Cache is unable to process the HTTP response header message from the application Web server. 11 - header too big The HTTP response header message from the application Web server is too large.
<i>Expiration</i> 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 E-3](#) lists the common event log error messages.

Table E-3 Error Events

Message	Description
Startup Initialization Events	
Failed to start the server	Oracle Web Cache is unable to open listening ports.
Oracle Web Cache Cache failed to initialize	Oracle Web Cache is unable to initialize the cache server process.
The server could not initialize	Oracle Web Cache is 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 is unable to load the Oracle Web Cache Manager help files and/or icons.
Could not increase number of file/socket descriptors to <i>connections</i>	Oracle Web Cache is unable to support the number connections. The number of maximum incoming connections can be adjusted in the Resource Limit page (Administering Oracle Web Cache > Resource Limits) of Oracle Web Cache Manager. See Also: "Greater Than One Thousand Maximum Connections" on page 10-7
HTTPS Startup Initialization Events	
Oracle Wallet Failed to open at location <i>location</i> . Only Auto Login Wallets Supported.	The wallet specified in the location cannot be opened. See Also: "Wrong or Older Cached Content" on page 10-11
No Oracle Wallet Configured with SSL. Server Initialization Failed.	The <WALLET> line is not specified in the <i>webcache.xml</i> file. See Also: "Wallet Cannot Be Opened" on page 10-8
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.

Message	Description
UNIX Process Identity Events	
Failed to find User (<i>user</i>) in /etc/password Invalid User ID (<i>user</i>)	The current user cannot perform this operation. Only the root user or the user specified in the Process Identity page (Administering Oracle Web Cache > Process Identity) of the Oracle Web Cache Manager can perform this operation.
Failed to find Group (<i>group</i>) in /etc/group Invalid Group ID (<i>group</i>)	The group ID that the current user is a member of is not valid for this operation.
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.
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 (Administering Oracle Web Cache > Resource Limits) of Oracle Web Cache Manager.
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 (Administering Oracle Web Cache > Resource Limits) of Oracle Web Cache Manager.

Message	Description
Operational Events	
Invalid XLF Field Name: <code>xlfield</code>	The XLF field specified for the access log file is invalid. The XLF fields are specified in the Access Logs page (Administering Web Sites > Access Logs) of Oracle Web Cache Manager.
Too many session definitions	The number of allowed session definitions used for cacheability rules for pages with personalized attributes and/or session tracking has exceeded 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).
Application Web Server Events	
Unable to resolve the IP address of <code>ip_address</code> . 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 <code>ip_address</code> . 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 username 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.

Message	Description
<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.

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.

admin server process

An Oracle Web Cache process that provides administration, configuration, and monitoring capabilities.

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

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

cache miss

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

cache server process

An Oracle9iAS Web Cache process that manages the cache by providing connection management and request processing.

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 an 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 Definition (DTD)

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

Edge Side Includes (ESI)

A markup language to enable **partial page caching** of HTML fragments.

embedded URL parameter

Parameter information embedded in the URL of documents. Oracle Web Cache accepts requests that use the following characters as delimiters: ampersand (&), dollar sign (\$), or semi-colon (;)

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.

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 enables 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 until it is back online.

GET method

An HTTP request 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

An HTTP request method made up of a URL and a query string containing parameters and values. An example of an 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.

HTTP response header

After receiving and interpreting a request message, an application Web server responds with an HTTP response header message.

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

Oracle Web Cache function 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 round-trip 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.

partial page caching

A feature that enables Oracle Web Cache to independently cache and manage fragments of HTML documents. A template page is configured with **Edge Side Includes (ESI)** markup tags that tell Oracle Web Cache to fetch and include the HTML fragments. The fragments themselves are HTML files containing discrete text or other objects.

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 like "Hello, *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

An HTTP request 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/regex_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.

Secure Sockets Layer (SSL)

A protocol developed by Netscape Corporation. SSL is an industry-accepted standard for network transport layer security. SSL provides authentication, encryption, and data integrity, in a public-key infrastructure (PKI). By supporting SSL, Oracle Web Cache is able to cache pages for **HTTPS protocol** requests.

selector

Cacheability can be evaluated against the following attributes:

- **URLs** of documents
- **HTTP request method** of documents
- Body of an HTTP **POST method**

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

Expiration time, [invalidation](#) time, and removal time of a document.

The closer the removal time, the lower a document's validity.

The higher the validity level, the longer Oracle Web Cache serves 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.

XLF

See [Extended LogFile Format \(XLF\)](#).

XML

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

Index

Symbols

- . (period) symbol
 - regular expression, 6-4, 8-9, 8-13
- " (double quotes) symbol
 - regular expression, 8-10, 8-13
- \$ (dollar sign) symbol
 - embedded URL parameter, 2-7
 - regular expression, 6-4, 8-9, 8-13
- & (ampersand) symbol
 - embedded URL parameter, 2-7
 - regular expression, 8-10, 8-13
- * (asterisk) symbol
 - regular expression, 6-4, 8-9, 8-13
- ;(semi-colon) symbol
 - embedded URL parameter, 2-7
- < (less than sign) symbol
 - regular expression, 8-10, 8-13
- <!--esi--> tag, Edge Side Includes (ESI), D-17
- > (greater than sign) symbol
 - regular expression, 8-10, 8-13
- ? (question mark) symbol
 - regular expression, 6-4, 8-9, 8-13
- [] (brackets) symbol
 - regular expression, 8-9, 8-13
- \ (backslash) symbol
 - regular expression, 6-4, 8-9, 8-13
- ^ (caret) symbol
 - regular expression, 6-4, 8-9, 8-13
- { } (braces) symbol
 - regular expression, 8-9, 8-13
- ' (single quotes) symbol
 - regular expression, 8-10

Numerics

- 1024 port, 10-5
- 1100 port, 5-7, B-1
- 4000 port, 5-15, B-1
- 4001 port, 5-15, 8-3, B-1
- 4002 port, 9-2, B-1
- 7777 port, 5-5
- 80 port, 5-5

A

- Accept request-header field, 2-11
- Accept-Charset request-header field, 2-11
- Accept-Encoding request-header field, 2-11
- Accept-Language request-header field, 2-11
- access logs
 - configuring settings for, 8-31
 - described, 8-27
 - disabling, 8-32
 - examples, 8-29 to 8-30
 - format, 8-27
 - XLF fields, 8-27
- access_log file, B-2
- access_logyyyyymmdd, 8-31
- ACTION invalidation message element, 8-10, C-4
- Active Server Pages (ASP), 1-8, 2-7
- admin server process
 - described, 8-2
 - restriction, 10-4
- administration port, 5-15
- administrator user, 5-2, B-1
- ADVANCEDSELECTOR invalidation message element, 8-9, C-3

- application Web servers
 - capacity, 5-5
 - concurrent connections, 2-5
 - configuring for Oracle Web Cache, 5-5
 - failover, 1-12, 1-16
 - connection request threshold, 5-6
 - polling failed servers, 5-7
 - load, 2-5
 - load balancing
 - configuration, 7-2
 - described, 1-10
 - performance assurance, 2-5
 - performance monitoring, 9-7
 - session binding
 - configuring, 7-3
 - described, 1-14
- Application Web Servers page in Oracle Web Cache Manager, 5-5
- attempt tag, Edge Side Includes (ESI), D-14
- authoritative DNS server, 3-19

B

- backend failover, 1-12
- BASICSELECTOR invalidation message
 - element, 8-8, C-3
- bin directory, A-1
- binding, 1-14

C

- Cache Cleanup page in Oracle Cache Manager, 8-13
- cache hit, 1-4, 2-2
 - Server response-header field, 2-3
- cache memory
 - configuration, 5-8
 - troubleshooting, 10-4
- cache miss, 1-4, 2-2
 - Server response-header field, 2-3
- cache population, 2-2
- cache server process
 - described, 8-2
 - memory restriction, 10-4
- cache size configuration, 5-8

- cacheability rules, 6-6 to 6-11
 - default, 6-5
 - overview, 6-2
 - PDF documents, 6-5
- capacity
 - described, 5-5
 - troubleshooting, 10-10
- category cookies
 - described, 2-10
 - request and response value comparison, 2-10
- certificate, 1-18
- certificate authority (CA), 1-17
- choose tag, Edge Side Includes (ESI), D-10
- comment tag, Edge Side Includes (ESI), D-15
- Common Gateway Interface (CGI), 1-8, 2-7
- compression
 - configuring, 6-8
 - described, 1-19
- configuring
 - access logs, 8-31
 - application Web server settings, 5-5
 - cache connection limit, 5-12
 - cache memory, 5-8
 - cacheability rules, 6-6 to 6-11
 - cacheability rules for HTTP error codes, 6-10
 - cacheability rules for multiple versions of the same document
 - cookie values, 6-9, 6-16
 - HTTP request headers, 6-18
 - cacheability rules for partial page caching, 6-33
 - cacheability rules for personalized attributes, 6-19
 - cacheability rules for session tracking, 6-28
 - cacheability rules for session-encoded URLs, 6-10, 6-19
 - compression, 6-8
 - Edge Side Includes (ESI), 6-7, 6-33
 - event logs, 8-25
 - expiration rules, 6-8, 6-14
 - failover of application Web servers, 7-2
 - load balancing of application Web server, 7-2
 - partial page caching, 6-33
 - quick reference to procedures, 4-7
 - resource limits, 5-8
 - security settings, 5-2
 - session binding to an application Web server, 7-3

- connection limit
 - configuring, 5-12
 - troubleshooting, 10-7
- COOKIE invalidation message element, 8-10, C-3, C-5
- cookies
 - cacheability rules
 - documents with category cookies, 6-9, 6-16
 - documents with personalized attributes, 6-10, 6-19
 - documents with session cookies, 6-10, 6-28
 - category cookies for multiple versions of the same URL, 2-10
 - described, 1-14
 - personalized attributes, 2-12
 - session cookies
 - for session binding, 1-14

D

- deploying Oracle Web Cache
 - customer example, 11-1 to 11-7
 - for part of a Web site, 3-10
 - in a distributed network, 3-17
 - in a failover pair, 3-12
 - inside a firewall, 3-15
 - outside a firewall, 3-16
 - with HTTPS requests, 3-4
 - with multiple application Web servers, 3-9
 - with one application Web server, 3-2
- directory structure
 - bin directory, A-1
 - docs directory, A-1
 - invalidation directory, A-1
 - lib directory, A-1
 - logs directory, A-2
 - mesg directory, A-2
 - utl directory, A-2
- DNS server, 1-4
- docs directory, A-1
- dynamically generated content caching, 1-8
 - Active Server Pages (ASP), 1-8, 2-7
 - Common Gateway Interface (CGI), 1-8, 2-7
 - described, 2-7
 - Java Server Pages (JSP), 1-8, 2-7

- Java Servlets, 1-8, 2-7
- multiple versions of the same document, 2-8
- personalized attributes, 2-12
- personalized greetings, 2-12
- PL/SQL Server Pages (PSP), 1-8, 2-7
- session tracking, 2-14
- session-encoded URLs, 2-15

E

- Edge Side Includes (ESI)
 - <!--esi--> tag, D-17
 - attempt tag, D-14
 - choose tag, D-10
 - comment tag, D-15
 - examples
 - personalized greeting, 6-44
 - portal site, 6-36
 - Surrogate-Control response-header field, 6-46
 - except tag, D-14
 - exception and error handling, D-7
 - HTTP_ACCEPT_LANGUAGE variable, D-4
 - HTTP_COOKIE variable, D-4
 - HTTP_HOST variable, D-4
 - HTTP_REFERER variable, D-4
 - HTTP_USER_AGENT variable, D-4
 - include tag, D-9
 - otherwise tag, D-10
 - personalized greetings, 6-34
 - QUERY_STRING variable, D-4
 - remove tag, D-16
 - Surrogate-Capability request-header field, 2-2, 2-3
 - Surrogate-Control response-header field, 6-45
 - try tag, D-14
 - vars tag, D-18
 - when tag, D-10
- embedded URL parameters
 - \$ (dollar sign) symbol, 2-7
 - & (ampersand) symbol, 2-7
 - ;(semi-colon) symbol, 2-7
- error messages in event log, E-6

- event logs
 - configuring settings, 8-25
 - described, 8-23
 - error messages, E-6
 - examples of, 8-23 to 8-25
 - finding errors, 8-25
 - format, 8-23
 - information messages, E-2
 - warning messages, E-4
- event_log file, B-2
- except tag, Edge Side Includes (ESI), D-14
- expiration
 - concepts of, 2-4
 - performance assurance heuristics, 6-15
- expiration rules, 6-8, 6-14
 - by cache entry, 6-14
 - by document creation, 6-14
 - by HTTP Expires header, 6-14
- Expires header, 6-14

F

- failover
 - configuring, 7-2
 - connection request threshold, 5-6
 - described, 1-12
 - polling failed application Web servers, 5-7
- features, new
 - cache status in Server response-header field, xxx
 - cacheability selectors, xxix
 - compression improvements, xxx
 - Edge Side Includes (ESI), xxix
 - HTTPS protocol support, xxix
 - invalidation improvements, xxx
 - Secure Sockets Layer (SSL), xxix
 - Surrogate-Control header, xxx
- firewalls and Oracle Web Cache deployments, 3-16
- FoundationPersistentSessionID session, 6-29

G

- GET method, 6-3, 6-6
- GET with query string method, 6-3, 6-6
- grep command, 8-25

- group ID for Oracle Web Cache
 - administration, 5-4

H

- HEADER invalidation message element, 8-10, C-4
- HTTP error code cacheability rules, 6-10
- HTTP request headers, 2-10, 6-9
 - cacheability rules for, 6-9, 6-18
 - supported by Oracle Web Cache, 2-11
- HTTP request-header fields
 - Accept, 2-11
 - Accept-Charset, 2-11
 - Accept-Encoding, 2-11
 - Accept-Language, 2-11
 - If-Modified-Since, 2-2
 - Last-Modified, 2-2
 - Surrogate-Capability, 2-3
 - Surrogate-Control, 2-3
 - User-Agent, 2-11
- HTTP requests
 - wallet configuration, 5-17
- HTTP response-header fields
 - Server, 2-3
 - Set-Cookie, 7-4
 - Surrogate-Control, 6-45
- HTTP_ACCEPT_LANGUAGE variable, D-4
- HTTP_COOKIE variable, D-4
- HTTP_HOST variable, D-4
- HTTP_REFERER variable, D-4
- HTTP_USER_AGENT variable, D-4
- HTTPS requests, 3-4
 - administration port configuration, 5-15
 - certificate, 1-18
 - certificate authority (CA), 1-17
 - invalidation port configuration, 5-16, 8-4
 - Oracle Web Cache listening port, 5-7
 - overview, 1-16, 5-7
 - statistics monitoring port configuration, 5-16, 9-2
 - wallet, 1-18

I

- ID invalidation response attribute, 8-12, C-5
- If-Modified-Since request-header field, 2-2
- include tag, Edge Side Includes (ESI), D-9
- information messages in event log, E-2
- internal_admin.xml file, A-1
- internal.xml file, A-1
- invalidate.c file, 8-16
- Invalidate.java file, 8-16
- invalidating documents
 - database triggers, 8-16
 - HTTP POST messages, 8-6
 - invalidate.c, 8-16
 - Invalidate.java, 8-16
 - Oracle Web Cache Manager, 8-12
 - scripts, 8-16
- invalidation
 - concepts of, 2-4
 - described, 1-9
 - performance assurance heuristics, 8-14
 - port number, 8-3, 9-2
 - user, 5-2
- invalidation directory, A-1
- INVALIDATION invalidation message
 - element, C-3
- invalidation messages
 - ACTION element, 8-10, C-4
 - ADVANCEDSELECTOR element, 8-9, C-3
 - BASICSELECTOR element, 8-8, C-3
 - compatibility with release 1.0, 8-7
 - COOKIE element, 8-10, C-3, C-5
 - HEADER element, 8-10, C-4
 - INVALIDATION element, C-3
 - NAME attribute, 8-10, C-3, C-4, C-5
 - OBJECT element, 8-8, C-3
 - regular expression
 - . (period) symbol, 8-9, 8-13
 - " (double quotes) symbol, 8-10, 8-13
 - \$ (dollar sign) symbol, 8-9, 8-13
 - & (ampersand) symbol, 8-10, 8-13
 - * (asterisk) symbol, 8-9, 8-13
 - < (less than sign) symbol>, 8-10, 8-13
 - > (greater than sign) symbol, 8-10, 8-13
 - ? (question mark) symbol, 8-9, 8-13
 - [] (brackets) symbol, 8-9, 8-13

- \ (backslash) symbol, 8-9, 8-13
- ^ (caret), 8-9, 8-13
- { } (braces) symbol, 8-9, 8-13
- ' (single quotes) symbol, 8-10
- REMOVALTTL attribute, 8-10, C-4
- URI attribute, 8-8, C-3
- URIPREFIX attribute, 8-9, 8-13
- VALUE attribute, 8-10, C-3, C-4
- WCInvalidation.dtd, A-1, C-1
- invalidation port, 5-16, 8-4
- invalidation responses, 8-12, C-5
 - ID attribute, 8-12, C-5
 - NUMINV attribute, 8-12, C-5
 - RESULT element, 8-12
 - STATUS attribute, 8-12, C-5
 - syntax of, 8-8
 - WCInvalidation.dtd, A-1, C-1
- invalidator user, 5-2, B-1

J

- Java Server Pages (JSP), 1-8, 2-7
- Java Servlets, 1-8, 2-7
- JSESSIONID session, 6-29

L

- L4 (Layer 4) switches, 3-4
- L7 (Layer 7) switches, 3-10
- Last-Modified request-header field, 2-2
- lib directory, A-1
- load balancing
 - configuring, 7-2
 - described, 1-10
- load on Oracle Web Cache computer, 10-11
- Local timestamp conversion issue, 8-26, 8-31
- logs directory, A-2

M

- msg directory, A-2
- multiple versions of the same document, 2-8
 - cookie values, 2-10, 6-16
 - HTTP request headers, 2-10, 6-9, 6-18

N

- NAME invalidation message attribute, 8-10, C-3, C-4, C-5
- netstat command, 5-12
- new features
 - cache status in Server response-header field, xxx
 - cacheability selectors, xxix
 - compression improvements, xxx
 - Edge Side Includes (ESI), xxix
 - invalidation improvements, xxx
 - protocol support, xxix
 - Secure Sockets Layer (SSL), xxix
 - Surrogate-Control header, xxx
- NUMINV invalidation response attribute, 8-12, C-5

O

- OBJECT invalidation message element, 8-8, C-3
- Oracle Web Cache
 - admin server process, 8-2
 - administration, 1-19
 - benefits
 - cost savings, 1-6
 - high availability, 1-6
 - network traffic reduction, 1-6
 - performance, 1-5
 - scalability, 1-5
 - cache server process, 8-2
 - deploying
 - for part of a Web site, 3-10
 - in a distributed network, 3-17
 - in a failover pair, 3-12
 - inside a firewall, 3-15
 - outside a firewall, 3-16
 - with HTTPS requests, 3-4
 - with multiple application Web servers, 3-9
 - with one application Web server, 3-2
 - described, 1-2
 - dynamically generated content caching, 1-8

features

- administration, 1-19
 - backend failover, 1-12
 - compression, 1-19
 - invalidation, 1-9
 - load balancing, 1-10
 - performance assurance, 1-9
 - restricted administration, 1-16
 - Secure Sockets Layer (SSL), 1-16
 - security, 1-16
 - session binding, 1-14
 - static content caching, 1-8
 - surge protection, 1-10
 - performance monitoring, 9-5
 - population of the cache, 2-2
 - with Oracle9i Application Server, 1-2
- Oracle Web Cache Manager
 - administering Oracle Web Cache
 - administration port, 5-15
 - cache memory, 5-8
 - connection limit, 5-12
 - event logs, 8-25
 - invalidation port, 5-16
 - invalidating documents, 8-12
 - resource limits, 5-8
 - security settings, 5-2
 - statistics monitoring port, 5-16
 - administering Web sites
 - access logs, 8-31
 - application Web server settings, 5-5
 - cacheability rules, 6-6 to 6-11
 - cacheability rules for HTTP error codes, 6-10
 - cacheability rules for multiple versions of the same document, 6-9, 6-16, 6-18
 - cacheability rules for personalized attributes, 6-19
 - cacheability rules for session tracking, 6-28
 - cacheability rules for session-encoded URLs, 6-10, 6-19
 - compression, 6-8
 - Edge Side Includes (ESI) permission, 6-7
 - expiration rules, 6-8, 6-14
 - load balancing, 7-2

- session binding, 7-3
 - wallet location, 5-18
- applying changes, 10-12
- described, 4-2
- introduced, 1-19
- monitoring application Web servers
 - performance statistics, 9-7
- monitoring Oracle Web Cache
 - cache health, 9-3
 - performance statistics, 9-5
- navigator pane, 4-5
- restarting Oracle Web Cache, 5-21
- right pane, 4-6
- starting, 4-2
- starting Oracle Web Cache, 8-2
- status messages, 4-4
- Oracle*9i* Application Server with Oracle Web Cache, 1-2
- Oracle*HOME_NAME*WebCache service, 5-19, 8-3
- Oracle*HOME_NAME*WebCacheAdmin service, 5-19, 8-3
- Oracle*HOME_NAME*WebCacheMon service, 5-19
- otherwise tag, Edge Side Includes (ESI), D-10

P

- partial page caching
 - cacheability rules, 6-33
 - configuring, 6-33
 - described, 2-17
 - examples
 - personalized greetings, 6-44
 - portal site, 6-36
 - Surrogate-Control response-header field, 6-46
 - Surrogate-Control response-header field, 6-45
- PDF documents, 6-5
- performance assurance
 - described, 2-4
 - introduced, 1-9

- performance assurance heuristics, 1-9, 2-4
 - application Web server limit, 2-5
 - application Web server load, 2-5
 - concurrent connections, 2-5
 - expiration, 6-15
 - invalidation, 8-14
 - popularity, 2-5
 - validity, 2-5
- performance monitoring
 - application Web servers, 9-7
 - Oracle Web Cache, 9-5
- personalized attribute cacheability rules, 6-19
- personalized attributes
 - Edge Side Includes (ESI), 6-34
 - WEBCACHEEND HTML tag, 2-12, 6-20
 - WEBCACHETAG HTML tag, 2-12, 6-20
- personalized greetings. See personalized attributes
- PL/SQL Server Pages (PSP), 1-8, 2-7
- popularity, 2-5
- populating the cache, 2-2
- port conflicts, 10-2
- ports
 - 1024, 10-5
 - 1100, 5-7, B-1
 - 4000, 5-15, B-1
 - 4001, 5-15, 8-3, B-1
 - 4002, 9-2, B-1
 - 7777, 5-5
 - 80, 5-5
 - administration port, 5-15
 - invalidation, 5-16, 8-4
 - Oracle Web Cache listening port, 5-7
 - statistics monitoring, 5-16, 9-2
- POSIX 1003 extended regular expressions, 6-4
- POST method, 6-3, 6-6
- privileged ports, 10-5

Q

- QUERY_STRING variable, D-4

R

- Range request header, 6-6
- readme.examples.html file, 8-16, A-1
- readme.html file, A-2
- regular expression, 6-4
 - . (period) symbol, 6-4, 8-9, 8-13
 - " (double quotes) symbol, 8-10, 8-13
 - \$ (dollar sign) symbol, 6-4, 8-9, 8-13
 - & (ampersand) symbol, 8-10, 8-13
 - * (asterisk) symbol, 6-4, 8-9, 8-13
 - < (less than sign) symbol>, 8-10, 8-13
 - > (greater than sign) symbol, 8-10, 8-13
 - ? (question mark) symbol, 6-4, 8-9, 8-13
 - [] (brackets) symbol, 8-9, 8-13
 - \ (backslash) symbol, 6-4, 8-9, 8-13
 - ^ (caret) symbol, 6-4, 8-9, 8-13
 - { } (braces) symbol, 8-9, 8-13
 - ' (single quotes) symbol, 8-10
- REMOVALTTL invalidation message
 - attribute, 8-10, C-4
- remove tag, Edge Side Includes (ESI), D-16
- Resource Limits page in Oracle Web Cache Manager, 5-10, 5-12
- restarting Oracle Web Cache, 5-21
- RESULT invalidation response element, 8-12
- reverse proxy server, 1-2
- rules for creating cacheability rules, 6-2

S

- Secure Sockets Layer (SSL), 1-16
 - see HTTPS requests
- security, 1-16
- Security page in Oracle Web Cache Manager, 5-2
- security settings, 5-2
- Server response-header field, 2-3
- session binding
 - configuring, 7-3
 - described, 1-14
- session cookies
 - described, 2-14
 - request and response value comparison, 2-15

- session tracking
 - cacheability rules, 6-28
 - described, 2-14
 - serving the first request from the cache, 6-32
- session-encoded URLs
 - cacheability rules for, 6-10, 6-19
 - described, 2-15
- Set-Cookie response-header field, 7-4
- starting
 - Oracle Web Cache, 5-2
 - Oracle Web Cache Manager, 4-2
- startup failures, 10-2
- static content caching, 1-8
- statistics monitoring port, 5-16, 9-2
- STATUS invalidation response attribute, 8-12, C-5
- status messages in Oracle Web Cache Manager, 4-4
- surge protection, 1-10
- Surrogate-Capability request-header field, 2-2, 2-3
- Surrogate-Control response-header field, 6-45
 - content="ESI/1.0" control directive, 6-46
 - content="webcache/1.0" control directive, 6-46
 - max-age control directive, 6-45
 - no-store control directive, 6-45
 - no-store-remote control directive, 6-45

T

- top utility, 10-11
- troubleshooting
 - application Web server capacity, 10-10
 - cache memory, 10-4
 - caching PDF documents, 6-5
 - connection limit, 10-7
 - GMT to local timestamp, 8-26, 8-31
 - load, 10-11
 - Oracle Web Cache Manager, 10-12
 - port conflicts, 10-2
 - privileged ports, 10-5
 - startup failures, 10-2
 - wallet configuration, 10-8
 - wrong or older cached content, 10-11
- trusted subnet for Oracle Web Cache administration, 5-3
- try tag, Edge Side Includes (ESI), D-14

U

uptime utility, 10-11
URI invalidation message attribute, 8-8, C-3
URIPREFIX invalidation message attribute, 8-9, C-3
user ID for Oracle Web Cache administration, 5-4
User-Agent request-header field, 2-11
utl directory, A-2
utl_proc.sql script, 8-16
UTL_TCP Oracle supplied package, 8-16

V

validity, 2-5
VALUE invalidation message attribute, 8-10, C-3, C-4
vars tag, Edge Side Includes (ESI), D-18

W

wallet
 configuring, 5-17
 considerations for Windows operating systems, 5-19
 described, 1-18
 troubleshooting, 10-8
warning messages in event log, E-4
WCSinvalidation.dtd, A-1, C-1
Web caching
 benefits
 cost savings, 1-6
 high availability, 1-6
 network traffic reduction, 1-6
 performance, 1-5
 scalability, 1-5
 described, 1-4
webcachectl start command, 4-7, 5-2, 5-21
webcachectl status command, 4-7, 8-2
webcachectl stop command, 4-7, 5-21
webcachectl utility
 obtaining the status of Oracle Web Cache, 8-2
 restarting Oracle Web Cache, 5-21
 starting Oracle Web Cache, 8-2
 stopping Oracle Web Cache, 8-2
webcachectl utility syntax, 4-7

webcached executable, A-1
WEBCACHEEND HTML tag for personalized attributes, 2-12, 6-20
WEBCACHETAG HTML tag for personalized attributes, 2-12, 6-20
webcache.xml file, 10-3, A-2
when tag, Edge Side Includes (ESI), D-10

X

XLf fields for access logs, 8-27

