Oracle9*i*AS Web Cache

Administration and Deployment Guide

Release 2 (9.0.3)

August 2002

Part No. B10055-01

ORACLE

Oracle9*i*AS Web Cache Administration and Deployment Guide, Release 2 (9.0.3)

Part No. B10055-01

Copyright © 2002 Oracle Corporation. All rights reserved.

Primary Author: Deborah Steiner

Contributing Author: Helen Grembowicz

Graphic Designer: Valarie Moore

Contributors: Marcos Almeida, Jesse Anton, Christine Chan, Bikui Chen, Rishi Divate, Ajay Dsai, Joe Errede, Jay Feenan, Patrick Fry, Ric Goell, Hideaki Hayashi, Larry Jacobs, Lars Klevan, Wei Lin, Gary Ling, Martin Littlecott, Xiang Liu, Rajiv Mishra, Jordan Parker, Marcin Porwit, Charles Qi, Cyril Scott, Jiangping Shi, Parthiban Thilagar, Bill 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, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

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

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

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

Oracle is a registered trademark, and Oracle Store, Oracle8*i*, Oracle9*i*, Oracle9*i*AS Discoverer, PL/SQL, and SQL*Plus are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

This program contains third-party code from Henry Spencer and the PHP Group. Under the terms of the Henry Spencer copyright notice and the PHP license, Oracle is required to provide the following notices. Note, however, that the Oracle program license that accompanied this product determines your right to use the Oracle program, including the PHP software, and the terms contained in the following notices do not change those rights.

Regular Expression Rights and License Notice The Windows release of Oracle9*i*AS Web Cache includes source code from the regular expression (regex) directory of release 4.01 of the PHP source distribution from the PHP Group. The regex source code is a copyright of Henry Spencer and licensed from the PHP Group. The following applies only to the regex code which is compiled and linked in the `webcached.exe` binary.

Henry Spencer Copyright Notice

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

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

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

1. The author is not responsible for the consequences of use of this software, no matter how awful, even if they arise from flaws in it.

2. The origin of this software must not be misrepresented, either by explicit claim or by omission. Since few users ever read sources, credits must appear in the documentation.

3. Altered versions must be plainly marked as such, and must not be misrepresented as being the original software. Since few users ever read sources, credits must appear in the documentation.

4. This notice may not be removed or altered.

PHP License Agreement for regex Source Code

Copyright © 1999, 2000 The PHP Group. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name "PHP" must not be used to endorse or promote products derived from this software without prior permission from the PHP Group. This does not apply to add-on libraries or tools that work in conjunction with PHP. In such a case, the PHP name may be used to indicate that the product supports PHP.

4. The PHP Group may publish revised and/or new versions of the license from time to time. Each version will be given a distinguishing version number. Once covered code has been published under a particular version of the license, you may always continue to use it under the terms of that version. You may also choose to use such covered code under the terms of any subsequent version of the license published by the PHP Group. No one other than the PHP Group has the right to modify the terms applicable to covered code created under this License.

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

2 Caching Concepts

3 Cache Clustering

4 Deploying Oracle9*i*AS Web Cache

5 Configuration and Administration Tools Overview

Part II Configuration and Administration of Oracle9*i*AS Web Cache

6 Initial Setup and Configuration

7 Creating Caching Rules

8 Administering Oracle9*i*AS Web Cache

9 Monitoring Performance

10 Troubleshooting Oracle9*i*AS Web Cache Configuration

Part III Reference

A Oracle9*i*AS Web Cache Directory Structure

B Oracle9*i*AS Web Cache Default Settings

C Invalidation and Statistics Document Type Definitions

D Edge Side Includes Language

E Event Log Messages

F Using Oracle9*i*AS Web Cache with Third-Party Application Web Servers

Glossary

Index

List of Figures

List of Tables

Send Us Your Comments

Oracle9*i*AS Web Cache Administration and Deployment Guide, Release 2 (9.0.3)

Part No. B10055-01

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: iasdocs_us@oracle.com
- FAX: 650-506-7407 Attn: Oracle9*i* Application Server Documentation Manager
- Postal service:
  Oracle Corporation
  Oracle9*i* Application Server Documentation
  500 Oracle Parkway, M/S 2op3
  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 Oracle9*i*AS Web Cache to cache both static and dynamically generated content from one or more **origin servers**.

This preface contains these topics:

Intended Audience

Oracle9iAS Web Cache Administration and Deployment Guide is intended for Web site administrators who perform the following tasks:

- Web site administration
- Origin 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 Oracle9iAS Web Cache"

Chapter 1, "Introduction to Oracle9iAS Web Cache"

This chapter introduces the architecture, benefits, and main features of Oracle9*i*AS Web Cache.

Chapter 2, "Caching Concepts"

This chapter explains how Oracle9*i*AS Web Cache is populated with content, how that content maintains consistency, and how dynamically generated content is cached.

Chapter 3, "Cache Clustering"

This chapter explains the concept of a **cache cluster**—that is, how multiple instances of Oracle9*i*AS Web Cache can run as independent caches, with no interaction with one another.

Chapter 4, "Deploying Oracle9iAS Web Cache"

This chapter presents several scenarios for deploying Oracle9*i*AS Web Cache.

Chapter 5, "Configuration and Administration Tools Overview"

This chapter introduces the various administration tools of Oracle9*i*AS 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 Oracle9iAS Web Cache"

Chapter 6, "Initial Setup and Configuration"

This chapter describes the steps to initially configure Oracle9*i*AS Web Cache to begin caching content. It also describes configuration options for deployments with origin servers and multiple Oracle9*i*AS Web Cache servers.

Chapter 7, "Creating Caching Rules"

This chapter explains how to configure caching rules.

Chapter 8, "Administering Oracle9iAS Web Cache"

This chapter describes how to start and stop Oracle9*i*AS 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 Oracle9iAS Web Cache Configuration"

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

Part III, "Reference"

Appendix A, "Oracle9iAS Web Cache Directory Structure"

This appendix describes the installed Oracle9*i*AS Web Cache directory structure.

Appendix B, "Oracle9iAS Web Cache Default Settings"

This appendix describes the default settings for Oracle9*i*AS Web Cache.

Appendix C, "Invalidation and Statistics Document Type Definitions"

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 fragments.

Appendix E, "Event Log Messages"

This appendix describes the most common event log messages.

Appendix F, "Using Oracle9iAS Web Cache with Third-Party Application Web Servers"

This appendix describes how Oracle9*i*AS Web Cache works with third-party application Web servers.

Glossary

The glossary defines terminology used throughout this guide.

Related Documentation

For more information, see these Oracle resources:

- *Oracle9i Application Server Concepts Guide*
- *Oracle9i Application Server Administrator's Guide*
- *Oracle9i Application Server Performance Guide*
- *Oracle9i Application Server Security Guide*
- *Oracle9iAS Clickstream Intelligence Administrator's Guide*
- *Oracle9iAS Containers for J2EE JSP Tag Libraries and Utilities Reference*
- *Oracle9iAS Discoverer Configuration Guide*
- *Oracle9iAS Portal Configuration Guide*
- *Oracle9iAS Single Sign-On Administrator's Guide*
- *Oracle9iAS Wireless Getting Started and System Guide*
- 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 free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://otn.oracle.com/admin/account/membership.html`

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://otn.oracle.com/docs/index.htm`

To access the database documentation search engine directly, please visit

`http://tahiti.oracle.com`

For additional information, see:

- `http://rfc.net/rfc1421.html` for further information about password Base64 encoding
- `http://rfc.net/rfc1738.html` for further information about URL encoding
- `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://www.cs.utah.edu/dept/old/texinfo/regex/regex_toc.html` for **regular expression** syntax
- `http://www.ietf.org/` for information about the **Open Systems Interconnection (OSI)**
- `http://www.cookiecentral.com/` for further information about **cookies**
- `http://www.gnu.org/software/wget/wget.html` for further information about the WGET utility for pre-populating a cache
- `http://www.edge-delivery.org` for further information about the **Edge Side Includes (ESI)** language
- `http://www.xslt.com/` for complete information about XSLT

Conventions

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

- Conventions in Text
- Conventions in Code Examples
- Conventions for Microsoft 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**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle9i Database Concepts* Ensure that the recovery catalog and target database do *not* 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 `sqlplus` to open SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase italic monospace (fixed-width) font`	Lowercase italic monospace font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run `U`*`old_release`*`.SQL` where *`old_release`* 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.	`DECIMAL (`*`digits`*` [ , `*`precision`*` ])`
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`
|	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.	`{ENABLE | DISABLE}` `[COMPRESS | NOCOMPRESS]`

Convention	Meaning	Example
. . .	Horizontal ellipsis points indicate either: ■ 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	`CREATE TABLE ... AS subquery;` `SELECT col1, col2, ... , coln FROM employees;`
. . .	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.	`acctbal NUMBER(11,2);` `acct CONSTANT NUMBER(4) := 3;`
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/system_password` `DB_NAME = database_name`

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.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

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.
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Conventions for Microsoft Windows Operating Systems

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

Convention	Meaning	Example
Choose Start >	How to start a program.	To start the Database Configuration Assistant, choose Start > Programs > Oracle - *HOME_NAME* > Configuration and Migration Tools > Database Configuration Assistant.
File and directory names	File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (|), and dash (-). The special character backslash (\) is treated as an element separator, even when it appears in quotes. If the file name begins with \\, then Windows assumes it uses the Universal Naming Convention.	`c:\winnt"\"system32` is the same as `C:\WINNT\SYSTEM32`
`C:\>`	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the *command prompt* in this manual.	`C:\oracle\oradata>`

	The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to your Windows operating system documentation for more information on escape and special characters.	`C:\>exp scott/tiger TABLES=emp QUERY=\"WHERE job='SALESMAN' and sal<1600\"`
		`C:\>imp SYSTEM/`*`password`*
		`FROMUSER=scott TABLES=(emp, dept)`

HOME_NAME	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`*`HOME_NAME`*`TNSListener`

Convention	Meaning	Example
ORACLE_HOME and *ORACLE_BASE*	In releases prior to Oracle8*i* release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory that by default used one of the following names: ■ `C:\orant` for Windows NT ■ `C:\orawin95` for Windows 95 ■ `C:\orawin98` for Windows 98 This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level *ORACLE_HOME* directory. There is a top level directory called *ORACLE_BASE* that by default is `C:\oracle`. If you install Oracle9*i* release 1 (9.0.1) on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is `C:\oracle\ora90`. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this guide follow OFA conventions. Refer to *Oracle9i Database Getting Starting for Windows* for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.	Go to the *ORACLE_BASE*`\`*ORACLE_HOME*`\rdbms\admin` directory.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. 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 Corporation 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/`

Accessibility of Code Examples in Documentation 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.

Accessibility of Links to External Web Sites in Documentation This documentation may contain links to Web sites of other companies or organizations that Oracle Corporation does not own or control. Oracle Corporation neither evaluates nor makes any representations regarding the accessibility of these Web sites.

What's New in Oracle9*i*AS Web Cache?

This section describes the new features of Oracle9*i*AS Web Cache release 9.0 and provides pointers to additional information. New features information from previous releases is also retained to help those users migrating to the current release.

The following sections describe the new features:

New Features in Release 2 (9.0.2 and 9.0.3)

The new features for Oracle9*i*AS Web Cache in release 2 (9.0.2 and 9.0.2) include:

- **Automatically Restarting a Cache Server**

  The **auto-restart process** checks that the **cache server process** is running and automatically restarts it if it is not running.

  See Also: Task 6: Configure Auto-Restart Process Settings on page 6-12

- **Cache Hierarchy**

  You can deploy Oracle9*i*AS Web Caches in a **cache hierarchy** so that an Oracle9*i*AS Web Cache server caches content from another Oracle9*i*AS Web Cache.

  See Also: "Cache Hierarchies" on page 1-16

- **Cache Cluster**

  You can configure multiple instances of Oracle9*i*AS Web Cache to run as members of a cache cluster. Cache clusters provide failure detection and failover of Web caches, increasing the availability of your Web site.

  See Also: Chapter 3, "Cache Clustering"

- **Edge Side Includes (ESI) Enhancements**

  This release of ESI provides support for the following:

  - Embedded HTML content with the `<esi:inline>` tag
  - Custom environment variables with the `<esi:environment>` tag
  - Expanded `<esi:include>` tag support that provides attributes for specifying expiration, fragment time-out, and the **HTTP request method**. In addition, new elements enable you to specify the HTTP request body of a fragment and the HTTP request header field and value for Oracle9*i*AS Web Cache to use.

    See Also: Appendix D, "Edge Side Includes Language"

- **HTTPS Support Between Oracle9*i*AS Web Cache and Application Web Servers**

  In addition to **HTTPS protocol** support between browsers and Oracle9*i*AS Web Cache, you can configure Oracle9*i*AS Web Cache for HTTPS support between Oracle9*i*AS Web Cache and **origin servers**.

  See Also: "Secure Sockets Layer (SSL) Support" on page 1-27

- **Invalidation Propagation**

  Invalidation messages are propagated in a cache hierarchy whereby one Oracle9*i*AS Web Cache server acts as an origin server for another Oracle9*i*AS Web Cache server. They are also propagated in a cache cluster with multiple Oracle9*i*AS Web Cache servers.

  See Also:

  - "Invalidation Propagation" on page 2-7
  - "Example: Propagating Invalidation Requests Throughout a Cache Cluster" on page 8-41

- **Invalidation Preview**

  You can preview the list of documents to be invalidated.

  See Also:

  - "Invalidation Preview Request Syntax" on page 8-17
  - "Manual Invalidation Using Oracle9iAS Web Cache Manager" on page 8-49

- **Listing the Contents of a Cache**

  You can generate a list of the URLs of the most popular documents stored in the cache and a list of the URLs of all of the objects currently in the cache.

  See Also: "Listing the Contents of the Cache" on page 8-49

- **Oracle Enterprise Manager Support**

  You can use **Oracle Enterprise Manager** for monitoring Oracle9*i*AS Web Cache. Oracle Enterprise Manager provides a Web-based tool that enables you to view Oracle9*i*AS Web Cache status and performance metrics.

See Also:

- "Oracle Enterprise Manager for Metrics" on page 5-8
- *Oracle9i Application Server Administrator's Guide*

- **Proxy Server Support**

  In addition to application Web servers for internal sites, you can configure Oracle9*i*AS Web Cache to send cache misses to proxy servers for external sites protected by a firewall.

  See Also: "Task 9: Specify Settings for Origin Servers" on page 6-23

- **Site Support**

  You can configure Oracle9*i*AS Web Cache to cache and assemble dynamic content for one or more Web sites.

  See Also: "Site Support" on page 1-10

- **webcachectl Commands**

  `webcachectl` provides new commands for finer-grain control of the `auto-restart` process, `cache` server process, and **admin server process**.

  See Also: "webcachectl Utility for Process Administration" on page 5-9

New Features in Release 2.0

The new features for Oracle9*i*AS Web Cache in release 2.0 include:

- **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 Oracle9*i*AS Web Cache Manager interface, as well as allowing other third-party caches to use Oracle9*i*AS Web Cache cacheability attributes.

  See Also: "Configuring Caching Attributes in Response Headers" on page 7-66

- **Caching 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 Caching Rules" on page 7-8

- **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: "Cache Freshness and Performance Assurance" on page 2-4

- **Compression**

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

  See Also: "Configuring Caching Rules" on page 7-8

- **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 2-26
  - Appendix D, "Edge Side Includes Language"

- **HTTPS Support**

  Oracle9*i*AS Web Cache is able to cache pages for HTTPS requests.

  See Also: "Secure Sockets Layer (SSL) Support" on page 1-27

- **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 8-6

Part I

Getting Started with Oracle9*i*AS Web Cache

Part I provides an overview of Oracle9*i*AS Web Cache concepts, products, and tools.

This part contains the following chapters:

1

Introduction to Oracle9*i*AS Web Cache

This chapter describes the performance barriers faced by Web sites and introduces the technology that can provide a complete caching solution.

This chapter contains these topics:

What is the Big Picture for Caching?

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

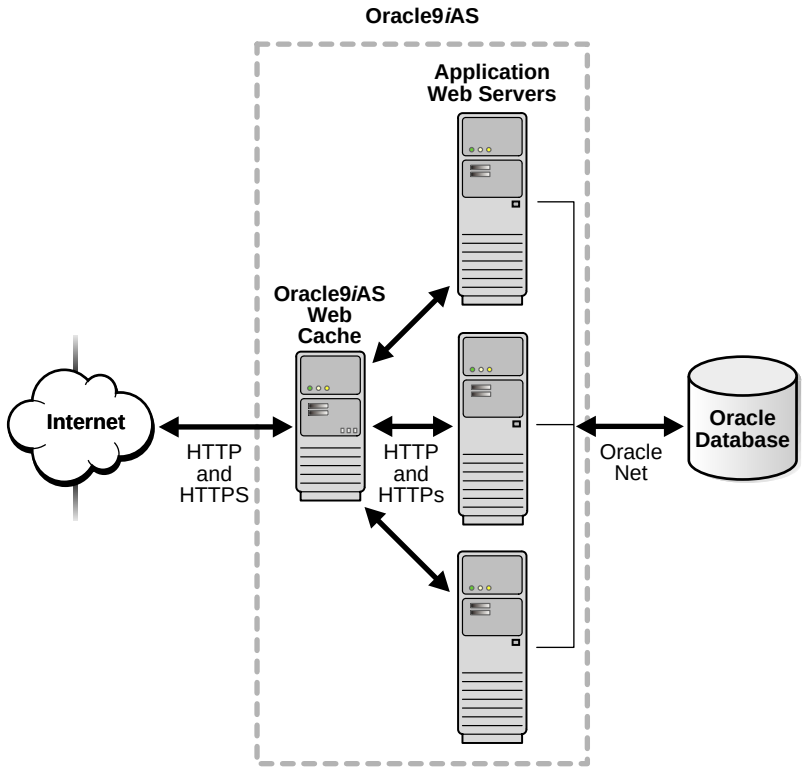
Oracle's Solution to Web Site Performance Issues

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 Oracle9*i*AS Web Cache to help e-businesses manage Web site performance issues. Oracle9*i*AS Web Cache is a content-aware server accelerator, or **reverse proxy server**, that improves the performance, scalability, and availability of Web sites that run on Oracle9*i* Application Server (Oracle9*i*AS) and Oracle9*i*.

By storing frequently accessed URLs in memory, Oracle9*i*AS Web Cache eliminates the need to repeatedly process requests for those URLs on the application Web server. Unlike legacy proxies that handle only static documents, Oracle9*i*AS Web Cache caches both static and dynamically generated content from one or more application Web servers. Because Oracle9*i*AS 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. Oracle9*i*AS 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 Oracle9*i*AS Web Cache. Oracle9*i*AS Web Cache, in turn, acts as a virtual server to the application Web servers. If the requested content has changed, Oracle9*i*AS 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 Oracle9iAS Web Cache Architecture



Note: Oracle9*i*AS Web Cache is compatible with Oracle HTTP Server or any other HTTP-compliant application Web server.

How Web Caching Works

To Web browsers, Oracle9*i*AS Web Cache acts as the virtual server for application Web servers. You configure a Load Balancer with the same IP address that is registered for a site's domain name and the application Web servers' host names. This Load Balancer receives requests for Oracle9*i*AS Web Cache. This configuration enables Web browsers to communicate with Oracle9*i*AS Web Cache rather than application Web servers when accessing a Web site.

Figure 1–2 on page 1-5 shows how Web caching works. Oracle9*i*AS 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 Oracle9*i*AS Web Cache follow:

1. A browser sends a request to a Web site named `www.company.com:80`.

   This request in turn generates a request to Domain Name System (DNS) for the IP address of the Web site.

2. DNS returns the IP address of the Load Balancer for the site, that is, `144.25.190.240`.

3. The browser sends the request for a Web page to the Load Balancer. In turn, the Load Balancer sends the request to Oracle9*i*AS Web Cache server `144.25.190.241`.

4. If the requested content is in its cache, then Oracle9*i*AS 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 Oracle9*i*AS Web Cache before being passed to the browser.

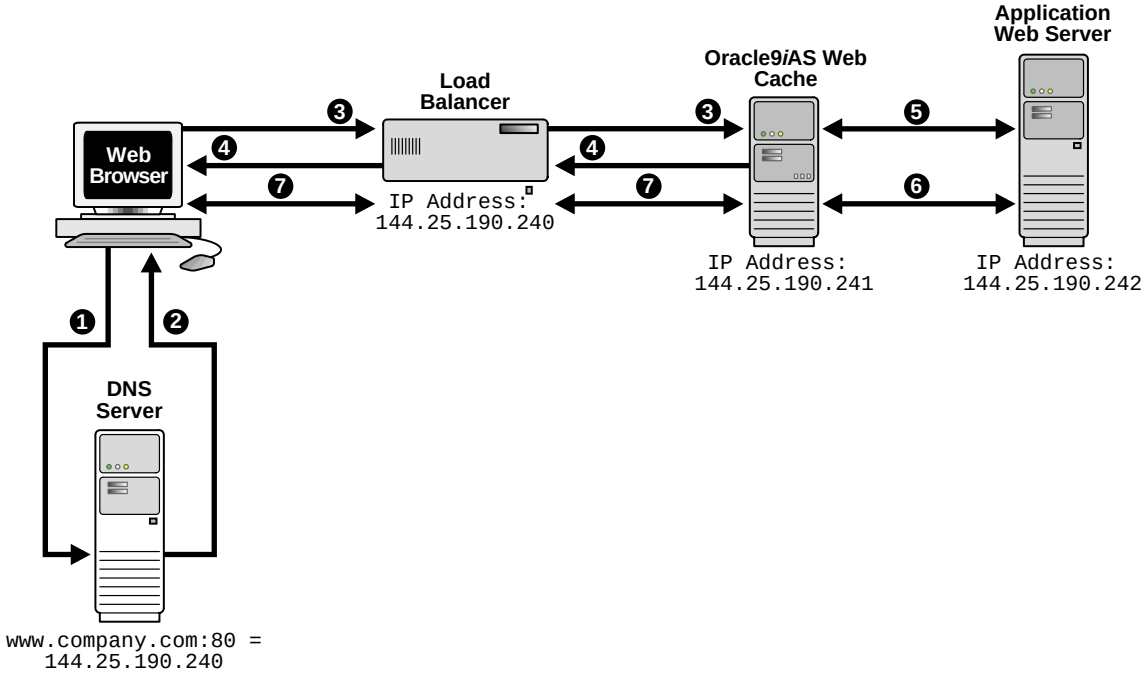
5. If Oracle9*i*AS Web Cache does not have the requested content or the content is stale or invalid, it hands the request off to application Web server `144.25.190.242`. This is called a **cache miss**.

6. The application Web server sends the content to Oracle9*i*AS Web Cache.

7. Oracle9*i*AS 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, Oracle9*i*AS Web Cache can increase the throughput of a Web site by several orders of magnitude. In addition, Oracle9*i*AS 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, Oracle9*i*AS 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

  Oracle9*i*AS Web Cache supports content-aware 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 the application Web server load increases.

- Cost Savings

  Better performance, scalability and availability translates into cost savings for Web site operators. Because fewer application Web servers are required to meet the challenges posed by traffic spikes and denial of service attacks, Oracle9*i*AS 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 Oracle9*i*AS 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 Oracle9*i*AS Web Cache

The main features of Oracle9*i*AS 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
- Site Support
- Cache Hierarchies
- Cache Clusters
- Application Web Server and Proxy Server Features
- Security Features
- Compression

Static and Dynamically Generated Content Caching

Caching rules determine which documents Oracle9*i*AS Web Cache caches. Rules fall into three categories:

- Rules for static content, such as GIF, JPEG, or static HTML files
- Rules for dynamically generated content created using technologies like 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 Oracle9*i*AS 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 **Edge Side Includes (ESI)** fragments

See Also:

- "Caching Dynamically Generated Content" on page 2-11 for further information about dynamically-generated content
- "Content Assembly and Partial Page Caching" on page 2-26

Cache Invalidation

Oracle9*i*AS Web Cache supports **invalidation** as a mechanism to keep the 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 Oracle9*i*AS Web Cache

  When documents are invalidated and a browser requests them, Oracle9*i*AS Web Cache refreshes them with new content from the application Web server.

- Assign an expiration time limit to the documents

  When a document expires, Oracle9*i*AS Web Cache treats it like an invalid document. If the document is requested by a browser, Oracle9*i*AS Web Cache refreshes it with 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, Oracle9*i*AS 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, then 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

Site Support

Oracle9*i*AS Web Cache caches and assembles dynamic content for one or more Web sites. From the perspective of Oracle9*i*AS Web Cache, a site can be either a **virtual host site** or an **ESI provider site**. Depending on the site category, you can configure Oracle9*i*AS Web Cache to perform different functions.

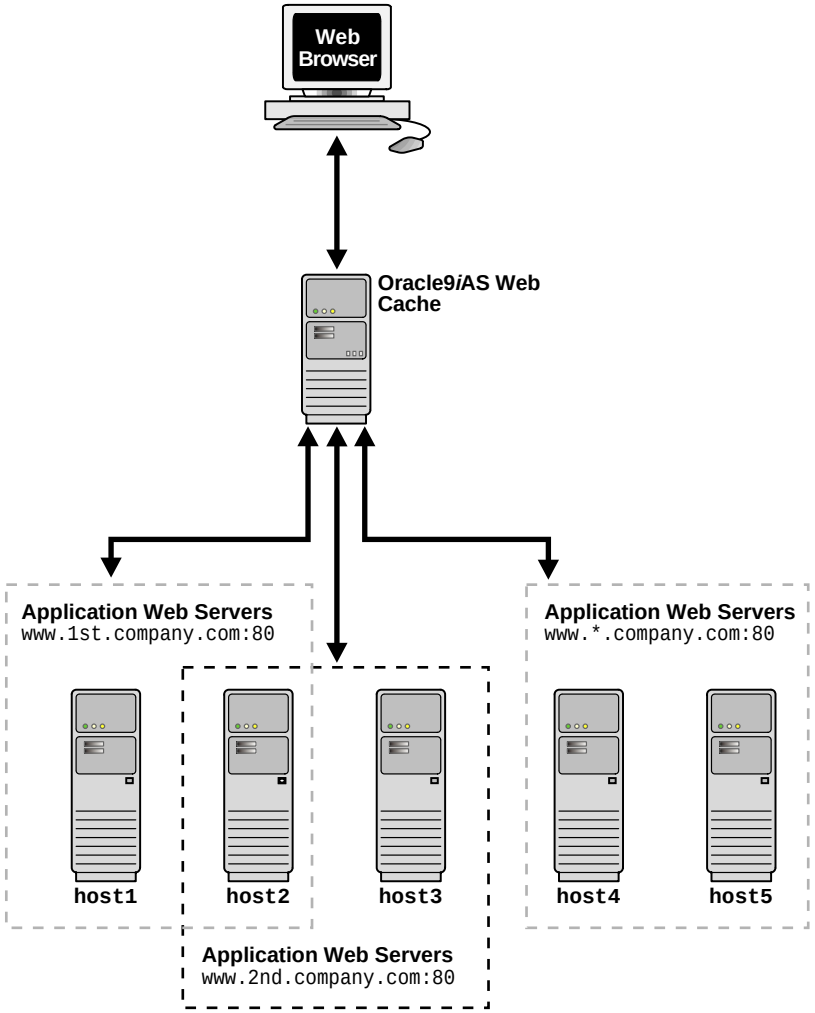
This section covers the following topics:

- Virtual Host Sites
- ESI Provider Sites
- Site Definitions and Site-to-Server Mappings
- How Oracle9iAS Web Cache Locates Application Web Severs or Proxy Servers

Virtual Host Sites

Virtual host sites are sites hosted by Oracle9*i*AS Web Cache. In other words, browsers can request cached content from these sites through Oracle9*i*AS Web Cache. Figure 1–3 on page 1-11 shows Oracle9*i*AS Web Cache caching content for two sites, `www.1st.company.com` and `www.2nd.company.com`. An additional mapping of `www.*.company.com` uses *, enabling additional virtual sites that map to `host4` and `host5` to be added. In addition to caching content, Oracle9*i*AS Web Cache can also assemble ESI fragments from these sites.

Figure 1–3 Multiple Virtual Host Sites



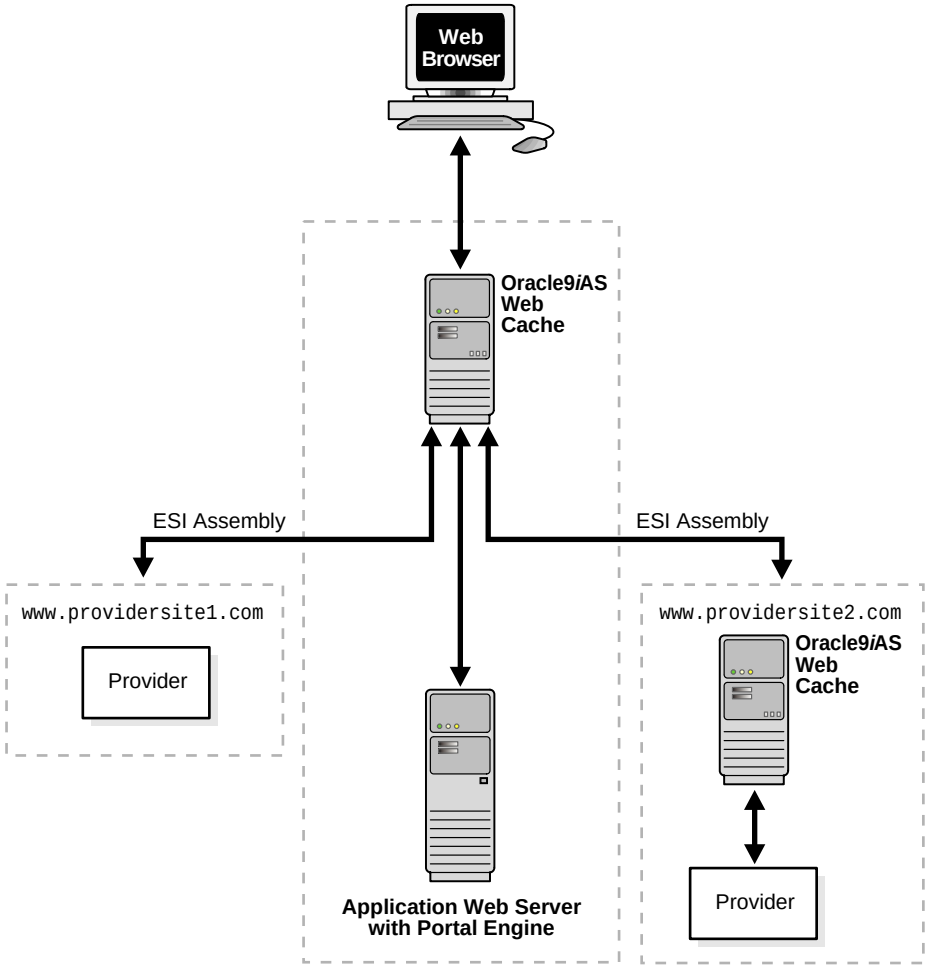
ESI Provider Sites

ESI provider sites are those sites that Oracle9*i*AS Web Cache contacts for ESI assembly only. Browsers are not allowed to request content from these sites.

Figure 1–4 on page 1-13 shows an ESI provider site configuration. In this configuration, Oracle9*i*AS Web Cache receives a request for a page with ESI markup tags. Oracle9*i*AS Web Cache sends the request to the application Web server. The application Web server uses a portal application to create a template page and sends it back to Oracle9*i*AS Web Cache for assembly. Oracle9*i*AS Web Cache includes the ESI fragments for the template page directly from the `www.providersite1.com` site and another Oracle9*i*AS Web Cache server, which is caching content for the `www.providersite2.com` site.

> **See Also:** "Content Assembly and Partial Page Caching" on page 2-26 for further information about ESI

Figure 1–4 Multiple ESI Provider Sites



Site Definitions and Site-to-Server Mappings

In order for Oracle9*i*AS Web Cache to recognize a virtual host site or an ESI provider site, administrators need to perform the following:

1. Specify a site definition:
   - Specify the name of the site
   - Specify the listening port number of the site
   - Specify the aliases for the site

   Many sites are represented by one or more aliases. Oracle9*i*AS Web Cache is able to recognize and cache requests for a site and its aliases. For example, site `www.company.com` may have an alias of `company.com`. By specifying this alias, Oracle9*i*AS Web Cache caches the same content from either `company.com` or `www.company.com`.
2. Determine if Oracle9*i*AS Web Cache communicates with application Web servers or **proxy servers**.

   Oracle9*i*AS Web Cache uses application Web servers for internal sites and proxy servers for external sites protected by a firewall.
3. Specify the host name and listening port number of the application Web server or proxy server.
4. Map the site to the application Web servers or proxy servers.
5. Create caching rules that apply to the site and global rules that apply to all sites.

   The site-specific caching rules are given a higher priority than the global rules.

Note: It may not be possible to specify a site definition for all external ESI provider sites. If an ESI request is made to a provider that does not match any application Web server mapping, then Oracle9*i*AS Web Cache uses Domain Name System (DNS) to resolve the site name. Note that this will not work if there is a firewall between the cache and the ESI provider. In that case, you must provide a proxy server mapping that directs the request to the appropriate proxy.

Undefined ESI provider sites disable the following Oracle9*i*AS Web Cache features:

- Performance assurance heuristics
- Application Web server and proxy server features, such as surge protection, load balancing, failover, and session binding

---

To further understand the mappings, reconsider Figure 1–3 on page 1-11. Web sites `www.1st.company.com:80` and `www.2nd.company.com:80` can have site aliases of `1st.company.com:80` and `2nd.company.com:80`, respectively. The site to application Web server mappings are as follows:

- `www.1st.company.com` maps to application Web servers `host1` and `host2`
- `www.2nd.company.com` maps to application Web servers `host2` and `host3`
- `www.*.company.com` maps to `host4` and `host5`

How Oracle9*i*AS Web Cache Locates Application Web Severs or Proxy Servers

When Oracle9*i*AS Web Cache receives a browser request for a document, it determines the destination site using one of the following elements:

- `Host` request-header field from the request
- Host portion of the requested URL
- `src` attribute of the ESI `<esi:include>` tag

Oracle9*i*AS Web Cache then looks up the configured site settings and mappings to determine if the site is supported, and the application Web servers or proxy servers and caching rules for the site. If there are no site settings and mappings for external ESI provider sites, Oracle9*i*AS Web Cache uses Domain Name System (DNS) to resolve the site name.

If the request does not include host information, then Oracle9*i*AS Web Cache sends the request to the default site. A default site is established for the Oracle HTTP Server when Oracle9*i* Application Web Server is installed. You can specify another site to be the default site.

See Also:

- "Using Oracle9iAS Web Cache to Support Multiple Sites"for deployment scenarios with application Web servers and proxy servers
- Chapter 6, "Initial Setup and Configuration" for configuration details
- "Default Site Settings" on page 6-33 for default site settings

Cache Hierarchies

For high availability and performance, many Internet businesses mirror their Web sites in strategic geographical locations. You can deploy Oracle9*i*AS Web Caches in a **cache hierarchy** so that an Oracle9*i*AS Web Cache server caches content from another Oracle9*i*AS Web Cache to a local market. Caches serving local markets shortens response time to these markets and reduces bandwidth and rack space costs for the content provider.

Oracle9*i*AS Web Cache provides supports two kinds of cache hierarchies:

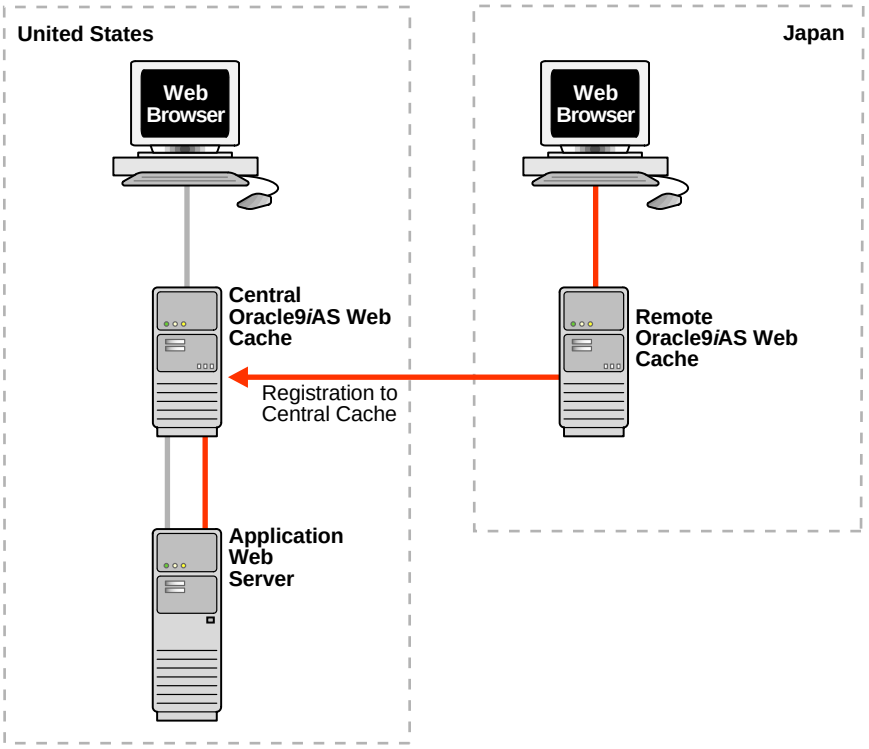
- A **distributed cache hierarchy** in which a **central cache** acts as an application Web server to a **remote cache**.
- An **ESI cache hierarchy** in which a **provider cache** acts as an application Web server to a **subscriber cache**.

A distributed cache hierarchy centralizes the management of application logic and data to the central cache and provides remote assembly and delivery of content. Compared with full-scale mirroring and database replication, a distributed cache hierarchy provides a more cost-effective model of distributed computing.

The remote cache is configured with the central cache as its application Web server. When the remote cache requests content from the central cache to serve a request, the remote cache identifies itself as an Oracle9*i*AS Web Cache to the central cache during a transparent registration process. Once registration is complete, the central cache establishes a hierarchical relationship with the remote cache. Registration also enables invalidation messages to be propagated from the central cache to the remote cache.

Figure 1–5 depicts a distributed cache hierarchy. A central cache resides in the United States office and a remote cache resides in the Japan. While the central cache in United States caches content from an application Web server, the remote cache in Japan caches content from the central cache.

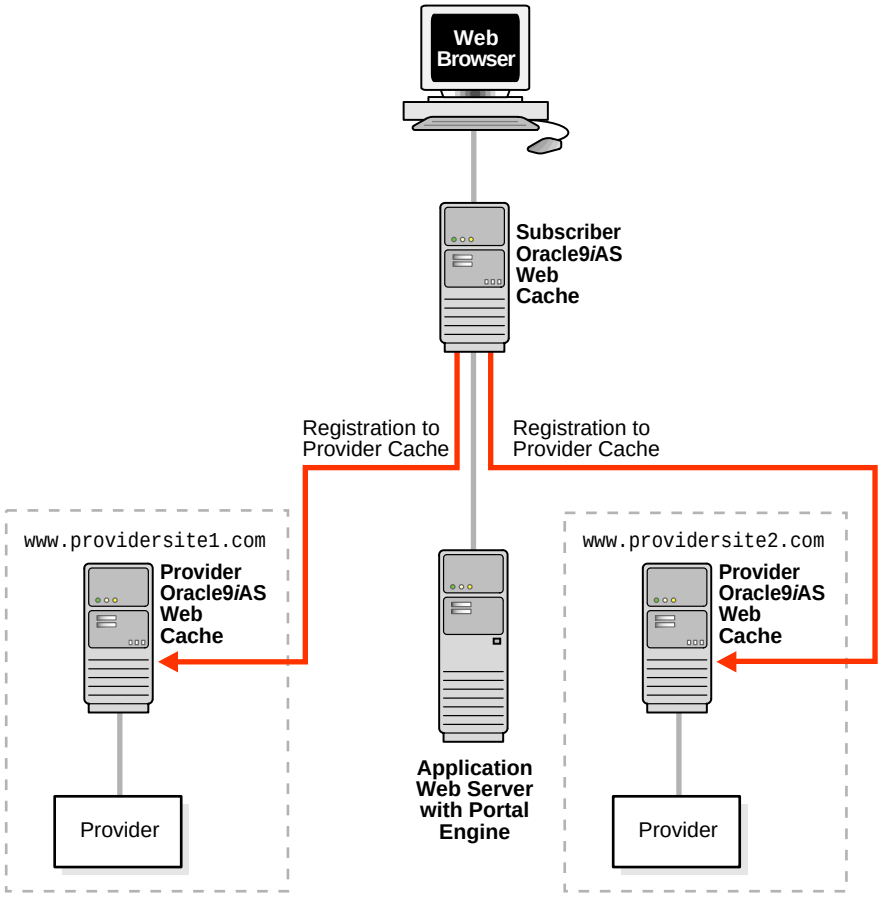
Figure 1–5 Distributed Cache Hierarchy



In an ESI cache hierarchy, the subscriber cache is configured with the provider caches as its application Web servers. When the subscriber cache requests content from the provider caches for ESI assembly, the subscriber cache identifies itself as an Oracle9*i*AS Web Cache to the provider caches during a transparent registration process. Once registration is complete, the provider caches establish a hierarchical relationship with the subscriber cache. Registration also enables invalidation messages to be propagated from the provider caches to the subscriber cache.

Figure 1–6 depicts an ESI cache hierarchy. A subscriber cache performs ESI assembly. Provider caches locally cache ESI fragments for **ESI provider sites** `www.providersite1.com` and `www.providersite2.com`. During ESI page assembly, the subscriber cache contacts the provider caches for the ESI fragments. By caching the ESI fragments locally on the provider caches, fragments are cached both by the provider and subscriber caches. This provides for quick page assembly.

Figure 1–6 ESI Cache Hierarchy



See Also:

- "Invalidation Propagation" on page 2-7 for information about how invalidation messages are propagated in a hierarchy of Oracle9*i*AS Web Cache servers
- "Configuring a Hierarchy of Caches" on page 6-48 for configuration details

Cache Clusters

To increase the availability and scalability of your Web site, you can configure multiple instances of Oracle9*i*AS Web Cache to run as members of a **cache cluster**. A cache cluster is a loosely coupled collection of cooperating Oracle9*i*AS Web Cache cache instances working together to provide a single logical cache.

Cache clusters provide failure detection and failover of caches, increasing the availability of your Web site. If a cache fails, other members of the cache cluster detect the failure and take over ownership of the cached content of the failed cluster member.

By distributing the Web site's content across multiple Web caches, more content can be cached and more client connections can be supported, expanding the capacity of your Web site.

See Also:

- Chapter 3, "Cache Clustering"for more information about cache clusters
- "Configuring a Cache Cluster" on page 6-55 for configuration details

Application Web Server and Proxy Server Features

Oracle9*i*AS Web Cache provides the following features for the application Web server and proxy server it supports:

- Surge Protection
- Load Balancing
- Backend Failover
- Session Binding

Where applicable, the term **origin server** is used in place of application Web server or proxy server to simplify the concepts presented in this section.

Surge Protection

Oracle9*i*AS Web Cache passes requests for non-cacheable or stale documents to the origin servers. To prevent an overload of requests on the origin servers, Oracle9*i*AS Web Cache has a surge protection feature that enables you to set a limit on the number of concurrent requests that the origin 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, Oracle9*i*AS Web Cache rejects the request and serves a site busy apology page to the Web browser that initiated the request.

Load Balancing

Most Web sites are served by multiple origin servers running on multiple computers that share the load of HTTP and HTTPS requests. All requests that Oracle9*i*AS Web Cache cannot serve are passed to the origin servers. Oracle9*i*AS Web Cache balances the load among origin servers by determining the percentage of the available **capacity**, the **weighted availability capacity** of each origin server. Oracle9*i*AS Web Cache sends a request to the origin server with the highest weighted available capacity. The weighted available capacity is determined by the following formula:

```
(Capacity - Load) / Capacity
```

where:

- `Capacity` is the maximum number of concurrent connections that the origin server can accept
- `Load` is the number of connections currently in use

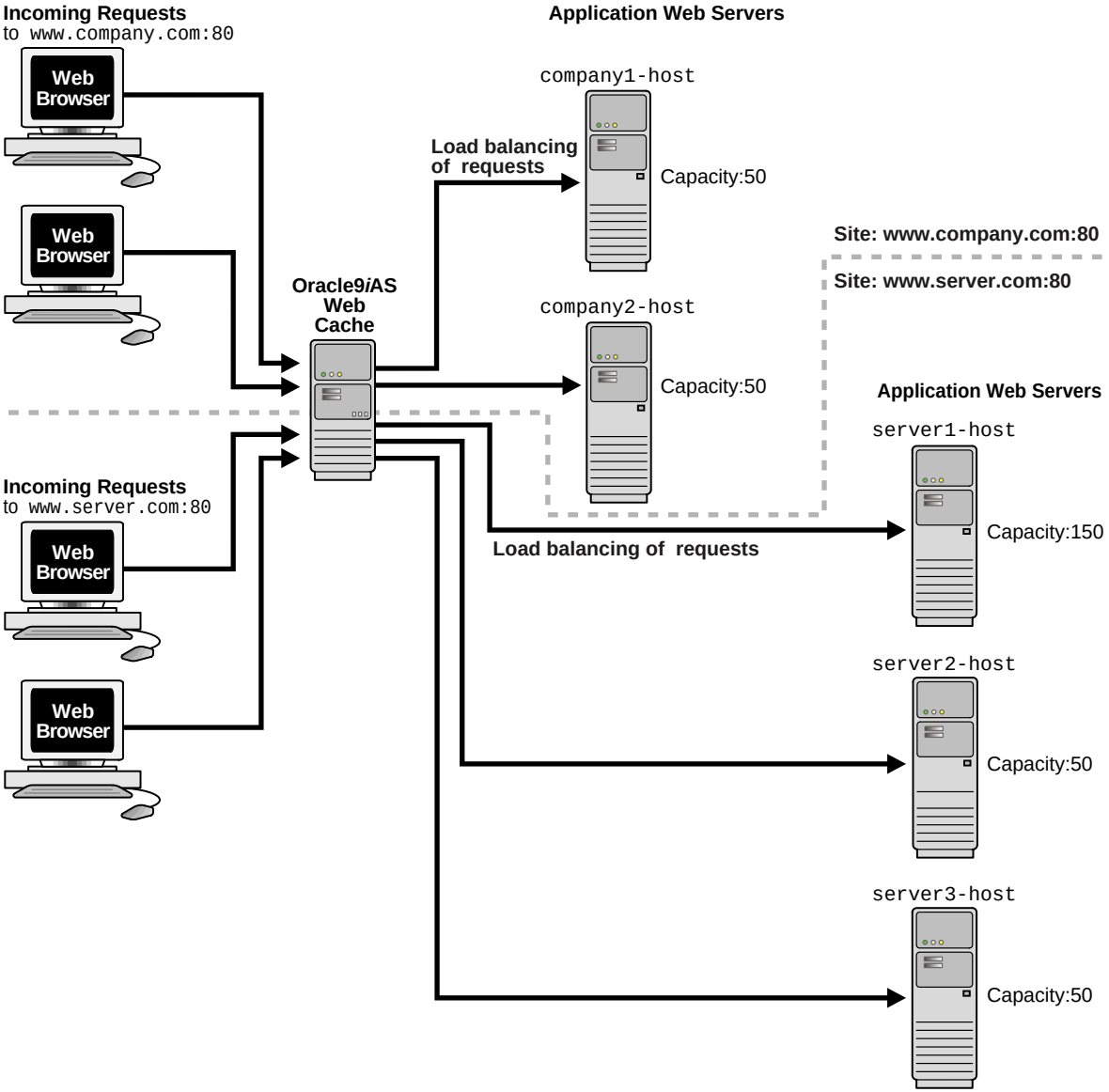
If the weighted available capacity is equal for multiple origin servers, then Oracle9*i*AS Web Cache sends requests to the origin servers in round-robin fashion. With round-robin, the first origin server in the list of configured servers receives the request, then the second origin server receives the second request.

If the weighted available capacity is not equal, Oracle9*i*AS Web Cache sends the request to the origin server with the highest weighted available capacity.

To configure load balancing for a site, set the capacity of each origin server, and create *one* site-to-server mapping that maps *all* the applicable origin servers to the site.

Figure 1–7 on page 1-22 shows two sites, `www.company.com:80` and `www.server.com:80`. `www.company.com:80` is supported by application Web servers `company1-host` and `company2-host` with capacities of 50 each. `www.server.com:80` is supported by application Web servers `server1-host`, `server2-host`, and `server3-host` with capacities of 150, 50, and 50, respectively.

Figure 1–7 Load Balancing



Assuming all application Web servers have an initial load of 0, the requests to `www.company.com:80` and `www.server.com:80` will be distributed in the following manner:

- The requests to `company1-host` and `company2-host` will be distributed between the two origin servers so that they maintain an equal load. The first request is sent to `company1-host`. The second request is sent to `company2-host` if `company1-host` is still processing the first request. The third and subsequent requests are sent to the origin server that has the highest weighted available capacity.
- The first request to `www.server.com:80` is sent to `server3-host`. The second request is sent to `server2-host`, because `server1-host` now has a weighted available capacity of 99.3 percent and `server2-host` has a weighted available capacity of 100 percent. The third request is sent to `server3-host` because `server2-host` now has a weighted available capacity of 98 percent and `server3-host` has a weighted available capacity of 100 percent. The fourth request is sent to `server1-host` because `server2-host` and `server3-host` now have weighted available capacities of 98 percent. The fifth request is sent to `server1-host` because its weighted available capacity is 98.6 percent, still greater than that of `server2-host` and `server3-host`.

See Also:

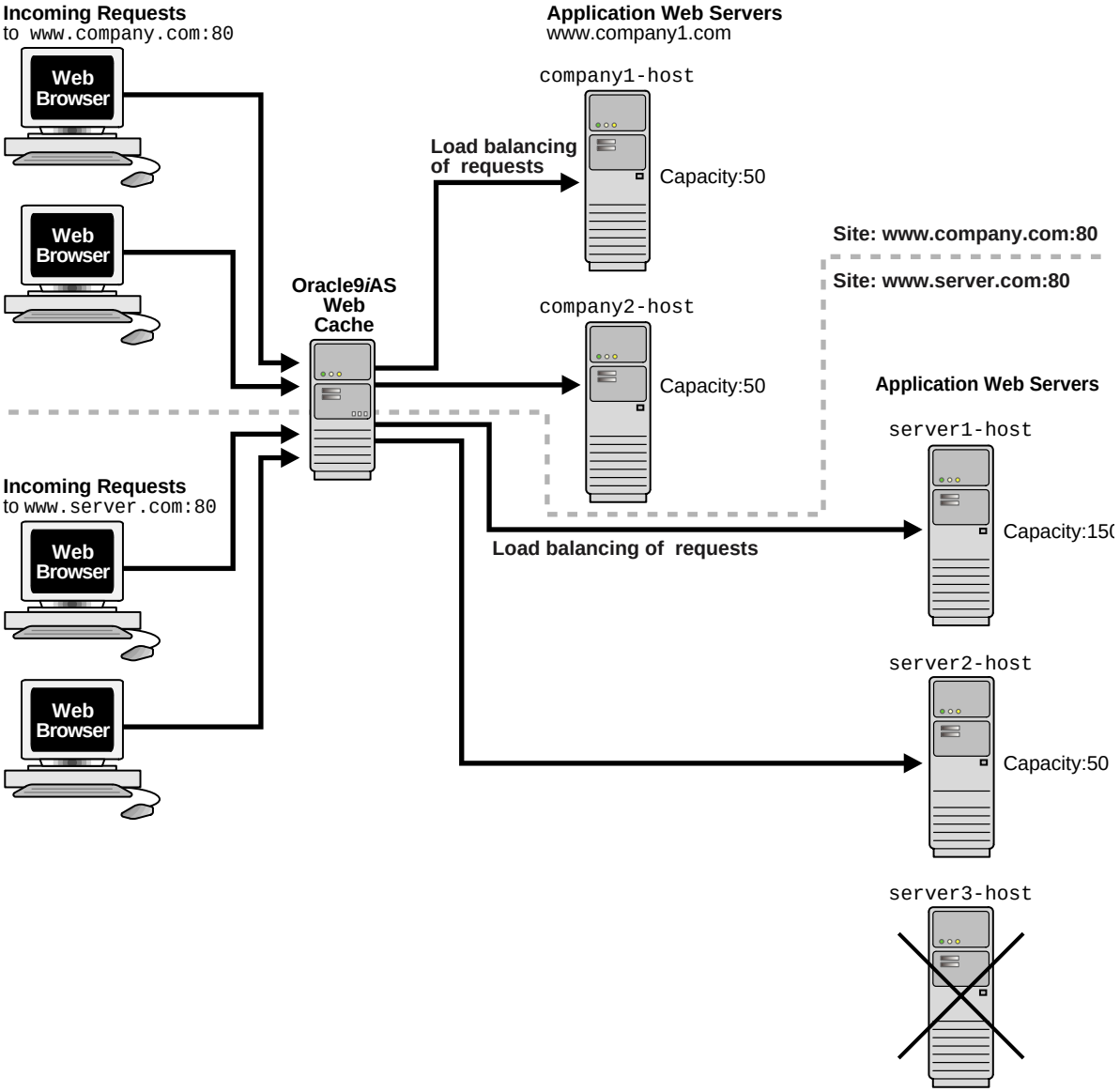
- "Task 9: Specify Settings for Origin Servers" on page 6-23 for application Web server or proxy server configuration
- "Configuring Load Balancing and Failover" on page 6-44 for load balancing and failover configuration

Backend Failover

After a specified number of continuous request failures, Oracle9*i*AS Web Cache considers an origin server as failed. When an origin server fails, Oracle9*i*AS Web Cache automatically distributes the load over the remaining origin servers based on the available load. Oracle9*i*AS Web Cache polls the failed origin server for its current up / down status until it is back online. When the failed server returns to operation, Oracle9*i*AS Web Cache will include its weighted available capacity to load balance requests.

The **failover** feature is shown in Figure 1–8 on page 1-24. An outage of `server3-host`, which had a capacity of 50, results in 75 percent of requests being distributed to `server1-host` and 25 percent request being distributed to `server2-host`.

Figure 1–8 Failover



Session Binding

Oracle9*i*AS Web Cache supports Web sites that use a session ID or **session cookie** to bind user sessions to a given origin server in order to maintain state for a period of time. To utilize the **session binding** feature, the origin server itself must maintain state, that is, it must be stateful. 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 1–9 on page 1-26 shows how Oracle9*i*AS Web Cache supports documents that use session binding:

1. When a request first comes in, Oracle9*i*AS Web Cache uses load balancing to determine to which origin server the request is forwarded. In this example, application Web server `www.server2.com` is selected.
2. If the requested document requires session binding, the origin server sends the session information back to the browser through Oracle9*i*AS Web Cache in the form of a cookie or an **embedded URL parameter**.
3. Oracle9*i*AS Web Cache sends subsequent requests for the session to the origin server that established the session, bypassing load balancing. In this example, application Web server `www.server2.com` handles the subsequent requests.

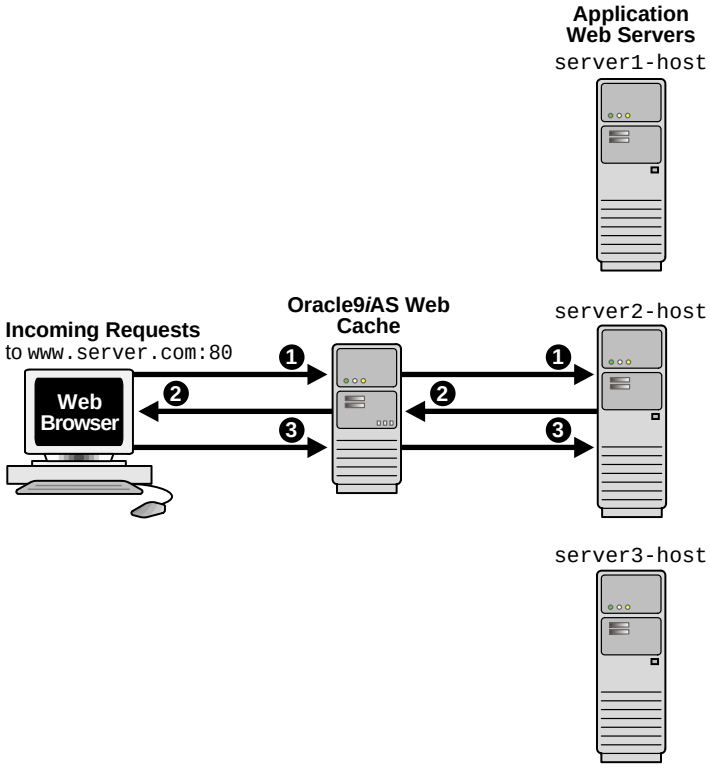
To configure session binding, specify a session definition that specifies the name of session cookie or embedded URL parameter.

> **See Also:** "Binding a Session to an Origin Server" on page 6-45 for configuration details

---

Note: If an origin server is busy, then Oracle9*i*AS Web Cache disables session binding to that origin server.

Figure 1–9 Session Binding



Security Features

Oracle9*i*AS Web Cache provides the following security-related features:

- Restricted Administration
- Secure Sockets Layer (SSL) Support

> **Note:** Oracle9*i*AS Web Cache does not cache pages that support basic HTTP authentication. These pages result in cache misses.

Restricted Administration

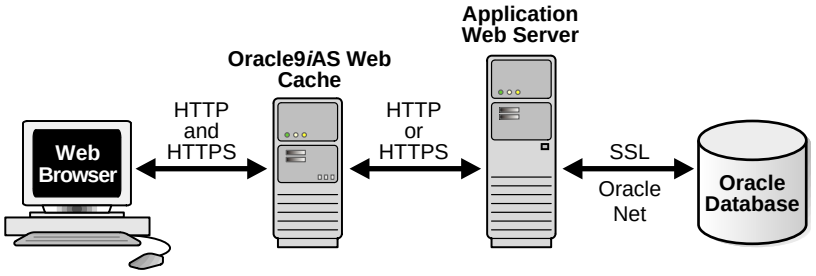
Oracle9*i*AS Web Cache restricts administration with the following features:

- 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, Oracle9*i*AS Web Cache is able to cache pages for HTTPS requests. As shown in Figure 1–10, you can configure Oracle9*i*AS Web Cache to receive HTTPS browser requests and send HTTPS requests to the origin servers.

Figure 1–10 SSL for Secure Connections



When sending requests to origin servers, note that HTTPS traffic can be processor intensive. If Oracle9*i*AS Web Cache needs to have traffic travel over the open Internet, then configure Oracle9*i*AS Web Cache to send HTTPS requests to the origin servers. If traffic only travels through a LAN in a data center, then the traffic can be sent with HTTP so as to reduce the load on the origin servers.

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

- Oracle9*i*AS Web Cache does not provide authentication or access control
- Oracle9*i*AS Web Cache does not support client-side certification

---

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 its private key. The Oracle9*i*AS 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 another 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

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 Oracle9*i*AS 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.

To support HTTPS for Oracle9*i*AS Web Cache, create a wallet on the Oracle9*i*AS Web Cache server for each supported site. When creating listening ports for Oracle9*i*AS Web Cache, specify the location of the wallet. One wallet can be shared among all the listening ports, or a separate wallet can be created for each port.

See Also:

- "Configuring Oracle9iAS Web Cache for HTTPS Requests" on page 6-38 for configuration details
- *Oracle9i Application Server Security Guide* for further information about Oracle Wallet Manager

How SSL Works To describe how SSL works in an HTTPS connection, the word client is used to describe either a browser or Oracle9*i*AS Web Cache, and the word server is used to describe either Oracle9*i*AS Web Cache or an origin server.

The authentication process between the client and server consists of the steps that follow:

1. The client initiates a connection to the server by using HTTPS.
2. SSL performs the handshake between the client and the server.

At the commencement of an HTTPS network connection between the client and server, an SSL handshake is performed. An SSL handshake includes the following actions:

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

Compression

You can select to have Oracle9*i*AS 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, Oracle9*i*AS Web Cache is able to compress text files by a factor of 4. For example, 300 KB files are compressed down to 75 KB.

Compatibility with Oracle9*i*AS Components

Table 1–1 describes Oracle9*i*AS Web Cache compatibility with other Oracle9*i*AS components.

Table 1–1 Oracle9iAS Web Cache Compatibility with Oracle9iAS Components

Oracle9*i*AS Component	Description
Oracle9*i*AS Clickstream Intelligence	Oracle9*i*AS Discoverer access logs import easily into Oracle9*i*AS Clickstream Intelligence, which provides a rich set of data warehousing and clickstream analysis functionality. **See Also:** "Configuring Access Logs" on page 8-64 and *Oracle9iAS Clickstream Intelligence Administrator's Guide*
Oracle9*i*AS Discoverer	Starting with Oracle9*i* Application Server release 2, Oracle9*i*AS Discoverer has been closely integrated with Oracle9*i*AS Web Cache to improve Discoverer Viewer's overall scalability, performance, and availability. Oracle9*i*AS Web Cache ships with a number of predefined caching rules for this purpose, and Oracle9*i*AS Discoverer uses ESI `Surrogate-Control` headers to govern cacheability of other non-configured responses. Because of this integration, the load on mid-tier and database servers in Oracle9*i*AS Discoverer deployments is reduced, more Discoverer Viewer users are able to access the system concurrently, and those users experience significantly better response times for workbook operations and common business intelligence queries. **See Also:** *Oracle9iAS Discoverer Configuration Guide*
Oracle9*i*AS Forms Services	Oracle9*i*AS Web Cache does not currently work applications that use Oracle9*i*AS Forms Services.
Oracle9*i*AS Portal	Oracle9*i*AS Web Cache has been closely integrated with Oracle9*i*AS Portal to improve Portal's overall scalability, performance and availability. Oracle9*i*AS Portal ships with a number of pre-defined caching and invalidation policies that ensure optimal use of Oracle9*i*AS Web Cache. Oracle9*i*AS Web Cache controls have been built into the Oracle9*i*AS Portal administrative user interface and can also be specified by content providers through the Portlet Developer Kit (PDK). **See Also:** Oracle9*iAS Portal* online help and *Oracle9iAS Portal Configuration Guide*
Oracle9*i*AS Reports Services	Oracle9*i*AS Web Cache cannot be used to accelerate Oracle9*i*AS Reports Services in this release.

Table 1–1 (Cont.) Oracle9iAS Web Cache Compatibility with Oracle9iAS Components

Oracle9*i*AS Component	Description
Oracle9*i*AS Single Sign-On	Applications that use Oracle9*i*AS Single Sign-On can take advantage of Oracle9*i*AS Wireless. Both Oracle9*i*AS Web Cache and `mod_SSO`, a feature of Oracle HTTP Server, have been configured out of the box to ensure that single sign-on authentication requests are tunneled transparently through Oracle9*i*AS Web Cache. No additional configuration is required on the customer side. **See Also:** *Oracle9iAS Single Sign-On Administrator's Guide*
Oracle9*i*AS Wireless	Oracle9*i*AS Wireless is integrated with Oracle9*i*AS Web Cache to improve page rendering performance and scalability. It should be noted that Oracle9*i*AS Web Cache does not understand WAP and is not used by Oracle9*i*AS Wireless in the traditional sense in that the cache does not "front-end" the wireless server. Instead, the cache is used as a repository for post-transformed content; the wireless runtime determines what content needs to be inserted into the cache and when to expire content in the cache. Oracle9*i*AS Wireless, in this case, acts as a device adaptation cache rather than a reverse-proxy cache. Since markup content is cached using Oracle9*i*AS Wireless, the performance and scalability benefits are due to two factors—reduced device adaptation costs and significantly reduced adapter invocation costs. The savings in terms of device adaptation costs stem form the fact that content that can be shared across users and sessions is essentially transformed only once (for each logical device) from its Mobile XML format. Secondly, since the content is not generated every time by an adapter, the total adapter invocation cost is significantly educed for a site that has a large subset of cacheable pages. **See Also:** *Oracle9iAS Wireless Getting Started and System Guide*

2

Caching Concepts

This chapter explains how Oracle9*i*AS Web Cache is populated with content, how that content maintains consistency, and how dynamically generated content is assembled and cached.

This chapter contains these topics:

Populating Oracle9*i*AS Web Cache

Oracle9*i*AS Web Cache uses caching rules to determine which documents to cache. When caching 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, Oracle9*i*AS 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. Oracle9*i*AS Web Cache then sends the request to the **origin server**. This is a **cache miss**. If the requested document is specified as one of the documents to cache, then Oracle9*i*AS Web Cache caches the document for subsequent requests. For a subsequent request for the document, Oracle9*i*AS Web Cache serves the document from its cache to the browser. This is a **cache hit**.

Note the following cache population considerations for browser requests:

- Browser requests with an `If-Modified-Since` request-header field

  When a browser sends a **GET method** request with an HTTP `If-Modified-Since` request-header field for a cached document, Oracle9*i*AS 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 Oracle9*i*AS Web Cache serves the cached document to the browser. When the `Last-Modified` header does not exist, Oracle9*i*AS 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 Oracle9*i*AS Web Cache sends an HTTP `304 Not Modified` status code to the browser.

- Browser requests with a `Range` request header field

  When the first browser request for a multi-part document with an HTTP `Range` request-header field comes in, Oracle9*i*AS Web Cache sends the request to the origin server. The origin server serves the entire document to the browser, and Oracle9*i*AS Web Cache caches the entire document for the request. For a subsequent request for the document, Oracle9*i*AS Web Cache serves only the part requested from the browser.

Notes:

- You can pre-populate the cache using Web crawler freeware such as WGET to warm up the cache on restart or after bulk invalidation operations. See `http://www.gnu.org/software/wget/wget.html` for further information about WGET.
- When you stop Oracle9*i*AS Web Cache, all objects are cleared from the cache. In addition, all statistics are cleared.

Cache Freshness and Performance Assurance

Consistency and performance are crucial for the reliability of Oracle9*i*AS Web Cache. This section contains the following topics:

Invalidation and Expiration

The features **invalidation** and **expiration** ensure consistency between the cache and the origin servers.

With invalidation, an HTTP `POST` message is sent by specifying which documents to mark as invalid. An invalidation message is intended for less predictable, more frequently changing content. When documents are marked as invalid and a browser requests them, they are removed and then refreshed with new content from the origin servers. You can select to remove and refresh invalid documents immediately, or base the removal and refresh on the current load of the origin servers.

With expiration, documents are marked as invalid after a certain amount of time in the cache. Expirations are useful if it can be accurately predicated when content will change on an origin server or database. If an expired document contains an `ETag` request-header, Oracle9*i*AS Web Cache first checks with the origin server to see if the document has changed since it was last refreshed. If the document is still valid, then the origin server sends an HTTP `304 Not Modified` status code to Oracle9*i*AS Web Cache and the document is not refreshed. If the document is no longer valid, then the origin server returns a full response, with an updated version of the document.

See Also:

- Section 13.3 Validation Model of the HTTP / 1.1 specification available at `http://www.ietf.org/rfc/rfc2616.txt` for further information about the validation caching and the `ETag` request header
- "Invalidating Documents in the Cache" on page 8-6 for instructions on invalidating content
- "Configuring Expiration Rules" on page 8-6 for instructions for configuring expiration rules

Performance Assurance Heuristics

One could logically assume that widespread cache invalidation or expiration would negatively impact performance of the origin servers, resulting in the generation of HTTP `503 Server Unavailable` errors to browsers. For this reason, Oracle9*i*AS Web Cache intelligently serves some of the documents stale until the origin servers have the capacity to refresh them.

Oracle9*i*AS 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

  Validity is based on the expiration time, invalidation time, and removal time of an object.

  Oracle9*i*AS Web Cache calculates validity by comparing the current time relative to an object's expiration or invalidation time and the object's scheduled removal time. Prior to expiration or invalidation time, the object is considered valid. Between expiration or 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 or 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.

- Popularity

  Popularity is determined by:

  - The number of times the object has been requested since insertion into the cache
  - The number of recent requests for the object

- Total available capacity of origin servers

  The total available capacity is determined by the following formula:

  ```
  Total Capacity - Total Load
  ```

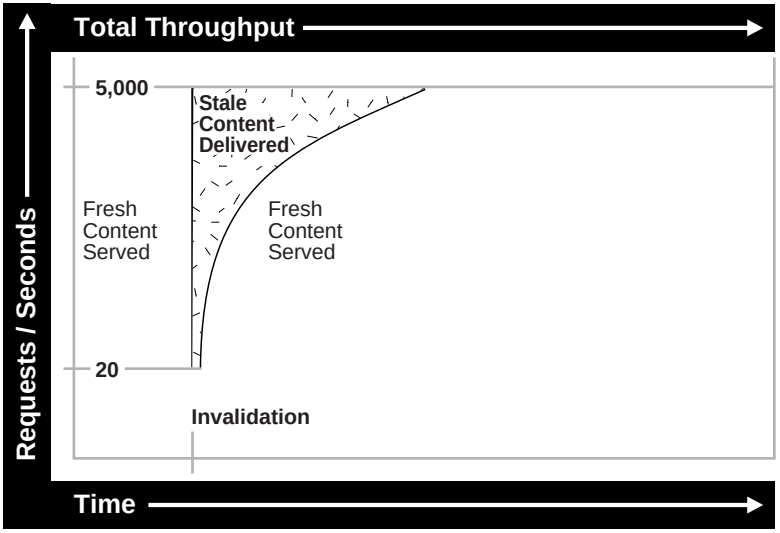
  In the formula:

  - `Total Capacity` is the accumulated maximum number of concurrent connections of all origin servers
  - `Total Load` is the total number of connections currently used by all the origin servers

Together, these factors provide Oracle9*i*AS Web Cache with a logical queue of content to update from the origin 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 to 5,000 documents for each second over a short period of time. This is because Oracle9*i*AS Web Cache refreshes the most popular documents first so that

these documents have little chance of being served stale. Once the popular documents are refreshed, the less popular documents are refreshed. The total number of documents that can revalidated in a given period of time is dependent on origin capacity. At the end of invalidation, only fresh content is served.

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

Invalidation Propagation

Propagation of invalidation messages from one Oracle9*i*AS Web Cache server to another occurs in the following deployments:

- Cache hierarchy whereby one Oracle9*i*AS Web Cache server acts as an origin server to another Oracle9*i*AS Web Cache server
- Cache cluster with multiple Oracle9*i*AS Web Cache servers

Invalidation in Hierarchies

In a configuration with a hierarchy of Oracle9*i*AS Web Cache servers, it is likely that content will be cached on multiple servers.

Figure 2–2 on page 2-8 depicts a **distributed cache hierarchy**. A **central cache** is located in the United States office, and a **remote cache** is located in the Japan office. While the central cache caches content from an application Web server, the remote cache caches content from the central cache. In other words, the central cache acts as an origin server to the remote cache in Japan.

When an invalidation message is sent to the central cache to refresh content, the central cache automatically propagates the invalidation message to the remote cache in Japan to ensure consistency.

Figure 2–2 Scenario 1: Invalidating Content in a Distributed Cache Hierarchy

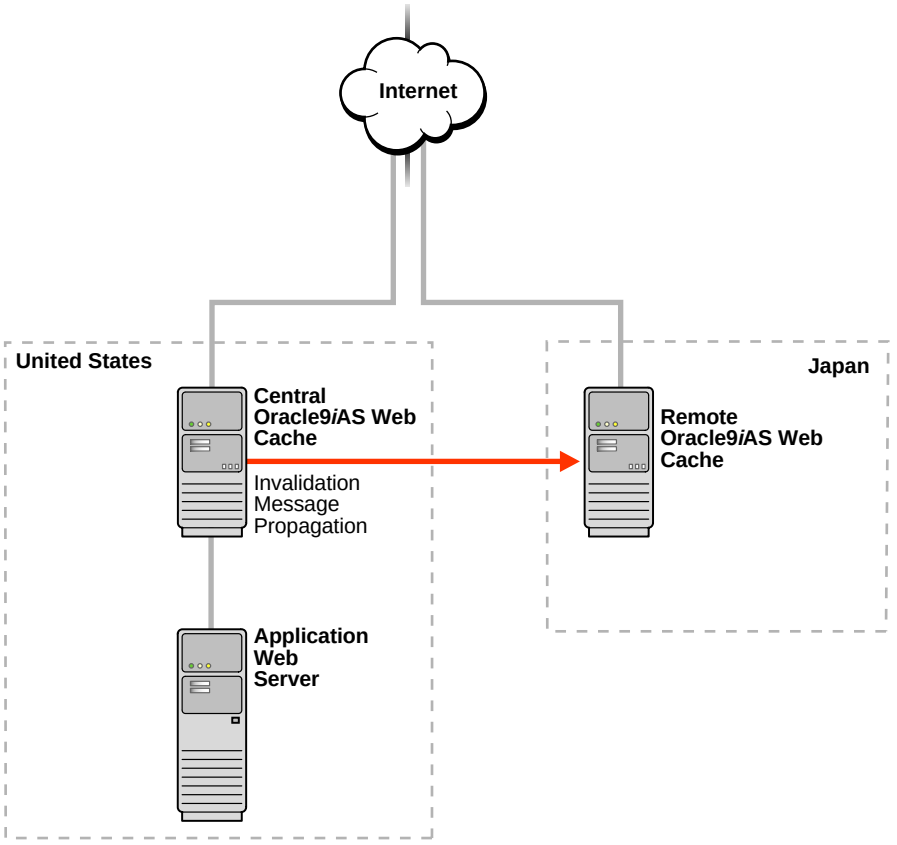
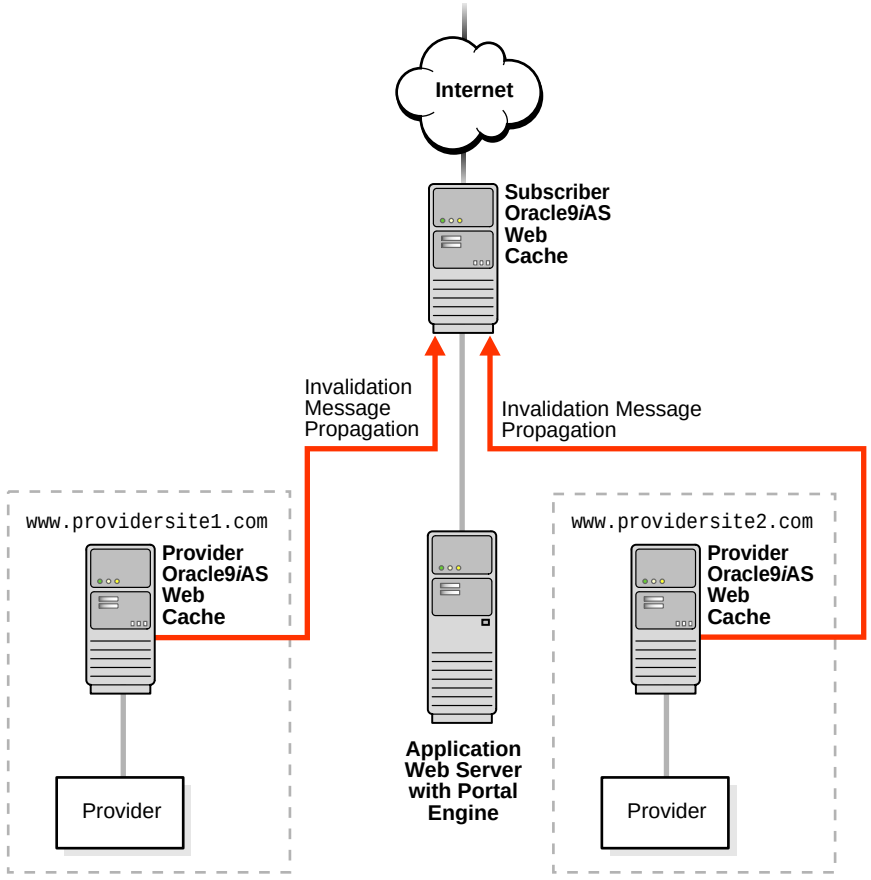


Figure 2–3 on page 2-9 depicts an **ESI cache hierarchy**. A **subscriber cache** performs **Edge Side Includes (ESI)** assembly. **Provider caches** locally cache ESI fragments for **ESI provider sites** `www.providersite1.com` and `www.providersite2.com`. During ESI page assembly, the subscriber cache contacts the provider caches for the ESI fragments.

Figure 2–3 Scenario 2: Invalidating Content in an ESI Cache Hierarchy



If content from the provider sites becomes invalid, an invalidation message is sent to the provider caches. In turn, the provider caches propagate invalidation message to the subscriber cache to ensure consistency.

To ensure that the a provider cache only invalidate its content, the subscriber cache checks the site host name specified in the invalidation message with the IP address of the provider cache from which the invalidation message propagated. If there is a match, then the subscriber cache processes the request. Otherwise, the request is rejected.

See Also:

- "Cache Hierarchies" on page 1-16 for an overview of cache hierarchies
- "Content Assembly and Partial Page Caching" on page 2-26 for an overview of ESI
- "Invalidating Documents in the Cache" on page 8-6 for instructions on invalidating content

Invalidation in Cache Clusters

In a cache cluster, administrators can decide whether to propagate invalidation messages to all **cache cluster members** or to send invalidation messages individually to cache cluster members.

When Oracle9*i*AS Web Cache propagates invalidation messages, it sends the invalidation messages to one cache cluster member who acts as the **invalidation coordinator.** The coordinator propagates the invalidation messages to the other cluster members. The coordinator waits for responses from all cluster members. When the propagation completes, it returns a message that lists, for each cluster member, the cluster member name, the status of the invalidation request, and the number of objects invalidated.

If the invalidation coordinator cannot be reached, Oracle9*i*AS Web Cache returns an error message and does not propagate the invalidation messages.

See Also: "Invalidating Documents in the Cache" on page 8-6 for instructions on invalidating content

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, because 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, Oracle9*i*AS Web Cache knows which documents are valid and which 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. Oracle9*i*AS Web Cache, on the other hand, receives invalidation messages from the origin server containing the original content.

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

The HTTP protocol has a way for browsers and origin servers to share information, such as session or category information, in message headers that browsers pass with every request to the origin server. This message header can contain a `Set-Cookie` response-header field that specifies a **cookie** and its value:

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

Cookies are stored on the browser's file system and are often used for identifying users who revisit Web sites. Browsers send a request with a `Cookie` request-header field with the cookie name and value that was received in the last response:

`Cookie:`*`cookie`*`=`*`value`*

Many users choose to disable cookies in their browsers out of privacy concerns. For this reason, applications often embed parameter information in the URL. Oracle9*i*AS Web Cache accepts requests that use the following characters as delimiters for **embedded URL parameters**: question mark (?), ampersand (&), dollar sign ($), or semi-colon (;).

Oracle9*i*AS Web Cache is able to recognize both cookies and embedded URL parameters, enabling it to recognize caching rules for documents with:

- Multiple Versions of the Same Document
- Personalized Attributes
- Session Information

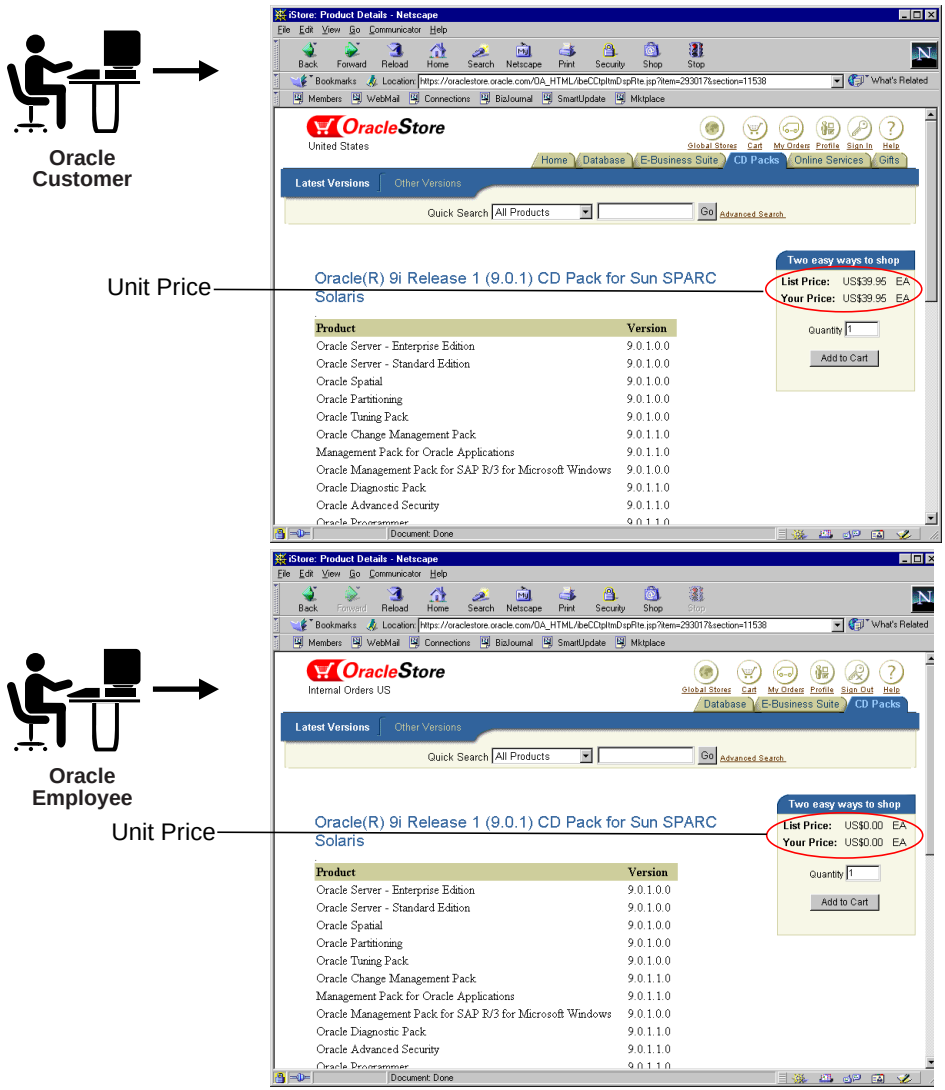
> **See Also:** `http://www.cookiecentral.com/` for further information about cookies

Note: The `Set-Cookie` response header field is not cached.

Multiple Versions of the Same Document

Some pages have multiple versions, enabling categorization. Figure 2–4 on page 2-13 shows the same document, `https://oraclestore.oracle.com/OA_HTML/ibeCCtpItmDspRte.jsp?item=293017&section=11538`, 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=INTERNAL`.

Figure 2–4 Multiple-Version Document



You can configure Oracle9*i*AS 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**), configure caching rules that specify the cookie name and whether to cache versions of the document that do not use the cookie.

When a browser sends an initial request for a multiple-version document, Oracle9*i*AS Web Cache passes the request to the origin server. In turn, the origin server includes a `Set-Cookie` response-header in the response with the category cookie and its value:

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

Upon receiving the `Set-Cookie` response-header field, the browser stores the cookie in memory. With its next request to the same origin server, the browser includes the `Cookie` request-header field with the category cookie name and value that was received in the last response:

`Cookie:`*`cookie=value`*

Oracle9*i*AS Web Cache evaluates whether the cookie and its value set in the `Set-Cookie` response-header matches the cookie and its value set in the `Cookie` request-header. If the cookie and value match, then the response is cached. If cookie and it value do not match, then the response is not cached. Once versions of the document are cached, Oracle9*i*AS Web Cache uses the value of the cookie in the browser's request to serve the appropriate version of the document to the browser.

Note: Oracle9*i*AS Web Cache does not cache the `Set-Cookie` response header field.

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 Oracle9*i*AS Web Cache to recognize the `user_type` cookie, enabling Oracle9*i*AS Web Cache to cache three different documents. In addition, you can configure Oracle9*i*AS 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	`http://www.dot.com/page1.htm`	`user_type=Customer`
2	`http://www.dot.com/page1.htm`	`user_type=Internal`
3	`http://www.dot.com/page1.htm`	`user_type=Promotional`
4	`http://www.dot.com/page1.htm`	No cookie

For those documents that use HTTP request headers, configure caching rules that specify the HTTP request header. HTTP request headers enable Web browsers to pass additional information about the request and about themselves. Oracle9*i*AS Web Cache uses the header to serve the appropriate version of the URL to browsers.

Oracle9*i*AS Web Cache supports all valid HTTP request headers. Table 2–2 lists the HTTP request-header fields supported by the Oracle9*i*AS Web Cache Manager interface, as described in "Configuring Caching Rules" on page 7-8. You can specify other HTTP request-header fields with the `Surrogate-Control` response-header field, as described in "Configuring Caching Attributes in Response Headers" on page 7-66.

Table 2–2 HTTP Request-Header Field

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`

Table 2–2 (Cont.) HTTP Request-Header Field

Header Field	Description
`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: Oracle9*i*AS Web Cache does not interpret the values of these HTTP request headers. If the values for two pages are different, Oracle9*i*AS 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 different versions of Internet Explorer, Oracle9*i*AS Web Cache serves two pages for the two requests.

Personalized Attributes

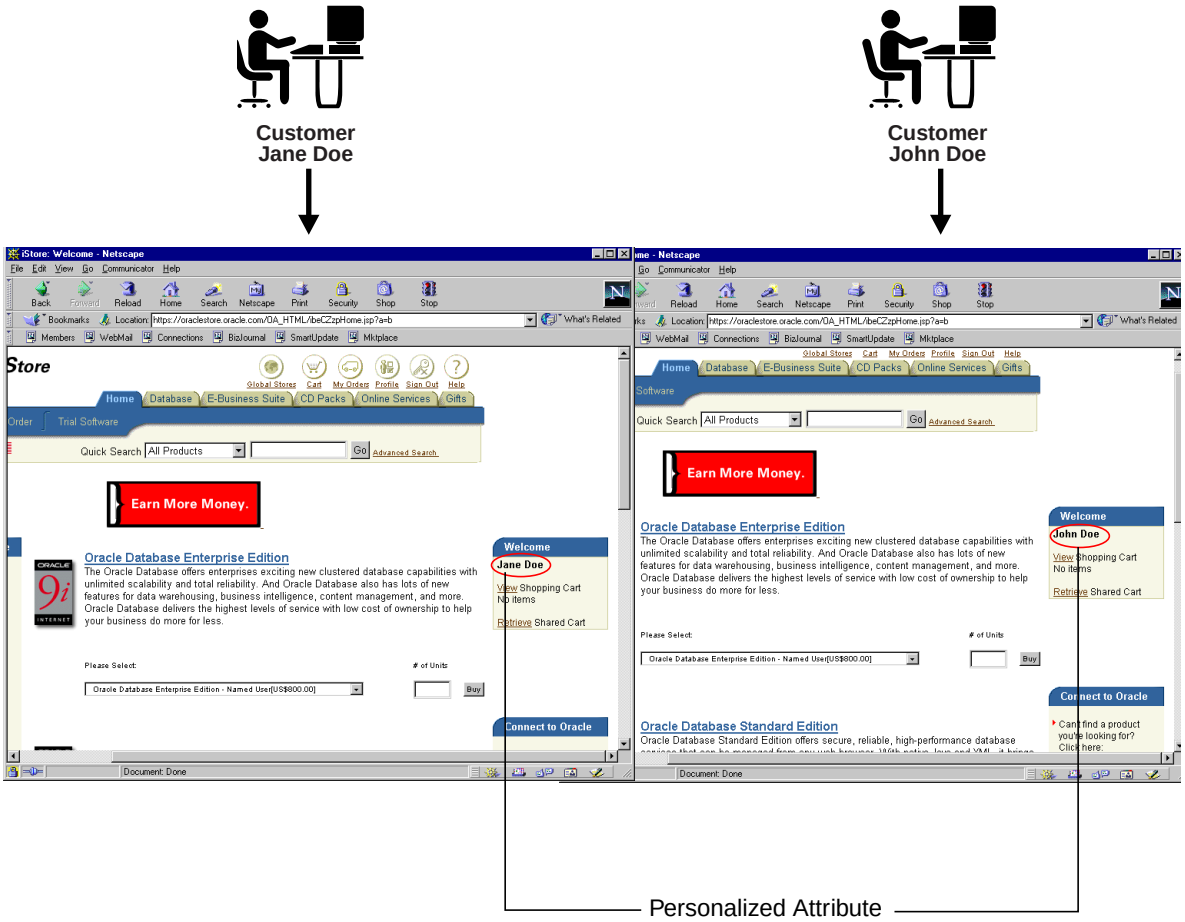
Many Web pages use a **personalized attribute** for personalized greetings like "Hello, *Name*," icons, addresses, or shopping cart snippets, on an otherwise generic page. You can mark the personalized attribute information with Oracle9*i*AS Web Cache HTML tags `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->`.

Oracle9*i*AS 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 Oracle9*i*AS Web Cache to use the same page for multiple users. Because only one page needs to be cached, only one origin 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 2–5 on page 2-18 shows two users, Jane Doe and John Doe, accessing the same page, `https://oraclestore.oracle.com/OA_HTML/ibeCZzpHome.jsp?a=b`. This page contains a personalized greeting suited for the user.

Figure 2–5 Page with a Personalized Attribute



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 personalized attribute definition assigned to the `person_name` cookie that Jane and John pass to Oracle9*i*AS 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 Oracle9*i*AS Web Cache receives the cookie information from Jane and John, it maps the `person_name` cookie to the `person01` personalized attribute definition 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 `https://oraclestore.oracle.com/OA_HTML/ibeCZzpHome.jsp?a=b&person_name=Jane+Doe`, and the page for John Doe could be `https://oraclestore.oracle.com/OA_HTML/ibeCZzpHome.jsp?a=b&person_name=John+Doe`. You could configure Oracle9*i*AS Web Cache with a personalized attribute definition of `person_name01` to map to the `person_name` embedded URL parameter. Oracle9*i*AS Web Cache would then use the value of the embedded parameter to substitute the appropriate name.

To substitute personalized attribute values:

1. Configure a personalized attribute definition with the personalized attribute cookie or embedded URL parameter.
2. Configure a caching rule with Simple Personalization enabled. Only requests matching the cacheability rule will perform the substitution.

If a request does not contain the cookie or embedded URL parameter, then Oracle9*i*AS Web Cache substitutes the personalized attribute with a default string.

See Also: "Configuring Personalized Attribute Definitions and Rules for Personalized Attributes" on page 7-28

Note: You can also substitute session values between the `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags. To substitute session values:

1. Configure a session definition with the session cookie or embedded URL parameter.
2. Configure a caching rule with Simple Personalization enabled. Only requests matching the cacheability rule will perform the substitution.

See Also: "Configuring Personalized Attribute Definitions and Rules for Personalized Attributes" on page 7-28

Controlling How Personalized Attribute Requests Are Served by the Cache

You can specify how Oracle9*i*AS Web Cache serves requests with the existence or nonexistence of personalized attribute cookies or embedded URL parameters. You can select to:

- Serve or not serve cached documents to requests that have a personalized attribute cookie or embedded URL parameter
- Serve or not serve cached documents to requests that do not have a personalized attribute cookie or embedded URL parameter

For example, if you want to require that the request get the personalized attribute cookie or embedded URL parameter settings from the origin server, then select to serve cached documents to requests that have the personalized attribute cookie or embedded URL parameter, but do not serve cached documents to requests that do not have the personalized attribute cookie or embedded URL parameter.

When you select to serve for both choices, you can then specify if requests with or without the personalized attribute cookie or embedded URL parameter can share the same cached document.

To specify how personalized attribute pages are served by Oracle9*i*AS Web Cache:

1. Configure a personalized attribute definition that specifies the name of the personalized attribute cookie or embedded URL parameter.
2. Specify the behavior for caching documents with or without personalized attribute information by defining a personalized attribute-related caching rule.
3. Associate URLs with the personalized attribute-related caching rule.

See Also: "Configuring Session-Related or Personalized Attributed-Related Caching Rules" on page 7-38

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. When a browser first accesses a Web site that uses session IDs, the origin server includes a `Set-Cookie` response-header in the response with the **session cookie** and value in order to establish a session:

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

Upon receiving the `Set-Cookie` response-header field, the browser stores the cookie in memory. With its next request to the same origin server, the browser includes the `Cookie` request-header field with the cookie name and value that was received in the last response:

`Cookie:`*`cookie=value`*

Because of the `Cookie` request-header field, the origin server determines that the browser already has a session and uses the value of the session to keep track of the browser state.

When returning a response to a request that already has a session, the origin server may or may not send a `Set-Cookie` header. If it does send it, it may or may not change the session cookie value. Origin servers really only need to send this response-header field to establish new cookies or change the value of the cookies.

Alternatively, origin servers can track a session with the browser by including the session value in an embedded URL parameter. With its next request to the same origin server, the browser includes the embedded URL parameter. Because of the embedded URL parameter, the origin server determines that the browser already has a session.

Using session information in a cookie or an embedded URL parameter, you can configure Oracle9*i*AS Web Cache for the following purposes:

- Ignoring the Value of Embedded URL Parameters
- Substituting Session Information in Session-Encoded URLs
- Controlling How Session Requests Are Served by the Cache

Ignoring the Value of Embedded URL Parameters

By default, Oracle9*i*AS Web Cache distinguishes origin server responses by the request URLs. However, if the request URL contains an embedded URL session parameter, then the request URL to the same page content is distinct for each session. Therefore, Oracle9*i*AS Web Cache will cache responses for each of the distinct URLs. This can result in low cache hit rates and redundantly cached documents.

You can configure Oracle9*i*AS Web Cache to ignore the value of embedded URL parameters so that one cached document is served to multiple sessions requesting the same page. Oracle9*i*AS Web Cache will then cache the response to the first request and serve subsequent requests for the page from its cache.

Note: Oracle9*i*AS Web Cache does not cache the `Set-Cookie` response header field.

Consider user Jane Doe accessing a page with a request URL of `https://oraclestore.oracle.com/OA_HTML/ibeCCtpSctDspRte.jsp?section=10103&session_ID=33436` and user John Doe requesting the same page with a request URL of `https://oraclestore.oracle.com/OA_HTML/ibeCCtpSctDspRte.jsp?section=10103&session_ID=33437`. The only distinct part of the URLs is the value of the `session_ID` parameter. Rather than caching and serving two versions of the same document, you can configure Oracle9*i*AS Web Cache to ignore the value of `session_ID` so that one cached document can be served to both users.

To ignore the value of URL parameters, configure a session definition in Oracle9*i*AS Web Cache that specifies the name of the session embedded URL parameter. Because you specify a session definition for a site or for all sites, the specified embedded URL parameter will be ignored for all requests to the relevant sites.

See Also: "Configuring Session Definitions to Exclude the Value of URL Parameters" on page 7-24

Substituting Session Information in Session-Encoded URLs

In section "Ignoring the Value of Embedded URL Parameters" on page 2-22, it is described how you can ignore the value of embedded URL parameters for documents with identical content for all sessions. However, in some cases, the HTML content of documents is programmed with hyperlink tags, such as `<A HREF=...>`, that contain embedded session information to distinguish users. These links are called **session-encoded URLs**. The use of session-encoded URLs results in responses that vary slightly from session to session.

You can configure Oracle9*i*AS Web Cache to substitute the values of session parameters in HTML hyperlink tags with the session information contained within a session cookie or an embedded URL parameter. Configuring session value substitution in combination with ignoring the value of embedded URL parameters, you can configure Oracle9*i*AS Web Cache to cache one document for multiple sessions, even if the session parameter values in session-encoded URLs vary.

Note: Oracle9*i*AS Web Cache does not cache the `Set-Cookie` response header field.

Continuing with the example from "Ignoring the Value of Embedded URL Parameters", assume that Jane Doe and John Doe are again assigned an embedded URL parameters of `session_ID=33436` and `session_ID=33437` by the origin server. The page shown in Figure 2–6 on page 2-24 has several `<A HREF=...>` links that include the `session_ID` parameter. The **Master Index** link under the **Oracle9i Release 1 (9.0.1)** heading for Jane Doe uses the following HTML code:

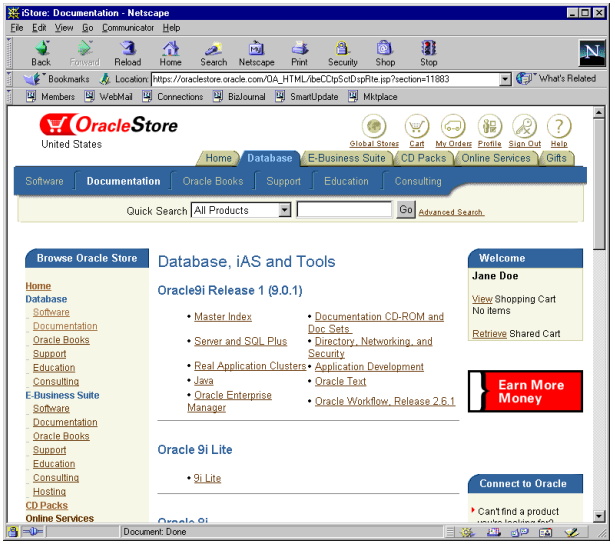
```
<A HREF="https://oraclestore.oracle.com/OA_
HTML/ibeCCtpSctDspRte.jsp?section=11886&session_ID=334326">Master
Index</A>
```

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

```
<A HREF="https://oraclestore.oracle.com/OA_
HTML/ibeCCtpSctDspRte.jsp?section=11886&session_ID=334327">Master
Index</A>
```

By using the value of the `session_ID` embedded URL parameter, Oracle9*i*AS Web Cache substitutes the correct session information for Jane Doe and John Doe.

Figure 2–6 Session-Encoded URLs



Once the cache is populated with a page that contains session-encoded URLs, other requests for the page are served from the cache, regardless of whether the request has a session cookie or embedded URL parameter. If the request does not contain a session cookie or embedded URL parameter, then Oracle9*i*AS Web Cache substitutes the session information in the session-encoded URLs with a configurable default string.

To substitute session values in session-encoded URLs:

1. Configure a session definition with the session cookie or embedded URL parameter. You can use the same session definition used for ignoring a URL parameter. When creating the session definition, configure the default string.
2. Configure a caching rule with Simple Personalization enabled. Only requests matching the caching rule will perform the substitution.

> **See Also:** "Configuring Session Definitions and Rules for Session-Encoded URLs" on page 7-26

Controlling How Session Requests Are Served by the Cache

You can specify how Oracle9*i*AS Web Cache serves requests with the existence or nonexistence of session cookies or embedded URL parameters. You can select to:

- Serve or not serve cached documents to requests that have a session cookie or embedded URL parameter
- Serve or not serve cached documents to requests that do not have a session cookie or embedded URL parameter

For example, if you want the first request of a new user to establish a session from the origin server, then select to serve cached documents to requests that have the session cookie or embedded URL parameter, but do not serve cached documents to requests that do not have the session cookie or embedded URL parameter.

When you select to serve for both choices, you can then specify if requests with or without the session cookie or embedded URL parameter can share the same cached document.

To specify how session-related pages are served by Oracle9*i*AS Web Cache:

1. Configure a session definition that specifies the name of the session cookie or embedded URL parameter.
2. Specify the behavior for caching documents with or without session information by defining a session-related caching rule.
3. Associate URLs with the session-related caching rule.

See Also:

- "Configuring Session-Related or Personalized Attributed-Related Caching Rules" on page 7-38 for configuration details
- "Configuring Caching Rules for Popular Pages with Session Establishment" on page 7-42 for information about caching popular pages that require session establishment

Content Assembly and Partial Page Caching

Oracle9*i*AS 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 caching profiles. These fragments are each maintained as separate elements in the origin server or content delivery network. The fragments are assembled into HTML pages as appropriate when requested by end users.

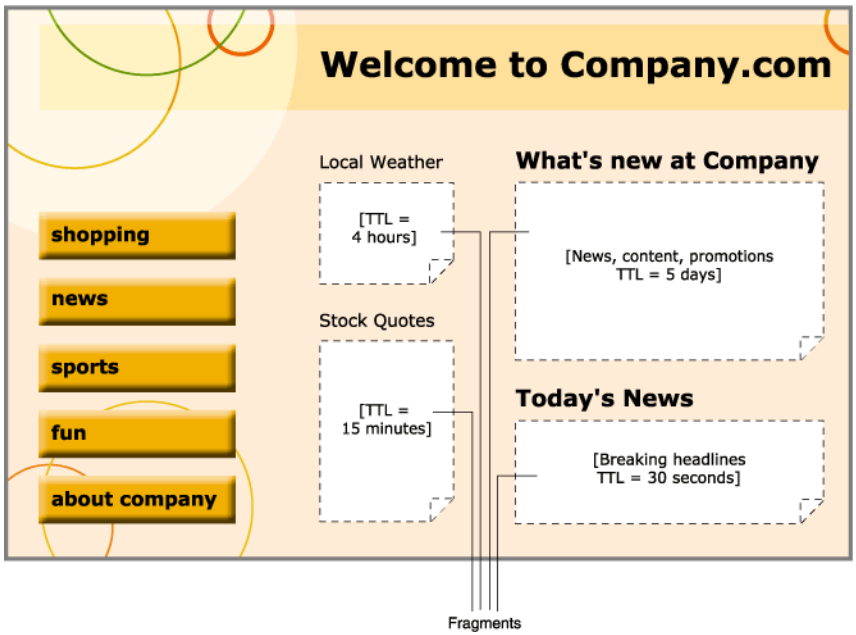
By enabling dynamic assembly of Web pages on Oracle9*i*AS Web Cache rather than on the origin servers, you can choose to cache some of the fragments of assembled pages. With **partial page caching**, much more HTML content can be cached, and then assembled and delivered by Oracle9*i*AS Web Cache when requested. Furthermore, page assembly can be conditional, based on information provided in HTTP request headers or end-user cookies.

- Page Assembly Components
- Fragmentation with the Inline and Include Tags
- Cookie Management for Template Pages and Fragments
- 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 fragments. As depicted in Figure 2–7, the template consists of common elements, such as a logo, navigation bars, framework, and other "look and feel" elements of the page. The fragments represent dynamic subsections of the page.

Figure 2–7 Template Page



The template page is associated with the URL that end users request. To include the fragments, the template page is configured with ESI markup tags that tell Oracle9*i*AS 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 caching 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
`<esi:choose>`	Performs conditional processing based on boolean expressions
`<esi:environment>`	Allows variable access from an HTTP response
`<esi:include>`	Includes an HTML fragment
`<esi:inline>`	Marks a fragment as a separately cacheable fragment, embedded in the HTTP response of another object
`<esi:remove>`	Specifies non-ESI markup if ESI processing is not enabled
`<esi:try>`	Specifies alternate processing when a request fails because the origin server is not accessible
`<esi:vars>`	Permits variable substitution for environment variables
`<!--esi...-->`	Specifies content to be processed

Figure 2–8 shows the ESI markup language for the template page shown in Figure 2–7 on page 2-27.

Figure 2–8 ESI Markup

```
<HTML>
<HEAD>
<TITLE>
Company.com
</TITLE>
</HEAD>
<BODY>
...
<!-- The following <esi:comment> tags are removed 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." />

 <esi:comment text="'Profile' refers to environment variables stored in
GetProfile.jsp. GetProfile.jsp enables access to 'PersonalInterest.' 'zipcode,'
'tickers,' and 'address' environment variables." />

 <esi:environment src="/GetProfile.jsp?sessionID=$(QUERY_STRING{sessionID})"
name="Profile" />
```

```
<esi:vars>
   <A HREF="/shopping.jsp?sessionID=$(QUERY_STRING{sessionID})">
     <IMG SRC="/img/shopping.gif">
   </A>
   <A HREF="/news.jsp?sessionID=$(QUERY_STRING{sessionID})">
     <IMG SRC="/img/news.gif">
   </A>
   <A HREF="/sports.jsp?sessionID=$(QUERY_STRING{sessionID})">
     <IMG SRC="/img/sports.gif">
   </A>
   <A HREF="/fun.jsp?sessionID=$(QUERY_STRING{sessionID})">
     <img src="/img/fun.gif">
  </A>
  <A HREF="/about.jsp?sessionID=$(QUERY_STRING{sessionID})">
     <iMG SRC="/img/about.gif">
   </A>
 </esi:vars>

 <esi:comment text="End: The quick link section" />
...
 <H3>Local Weather</H3>
 <esi:include src="/weather.jsp?sessionID=$(QUERY_
STRING{sessionID})&zipcode=$(Profile{zipcode})" />
...

 <H3>Stock Quotes</H3>
 <esi:try>
   <esi:attempt>
     <esi:include src="/CompanyStack.jsp?sessionID=$(QUERY_
STRING{sessionID})&tickers=$(Profiles{tickers})" />
   </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="$(Profile{PersonalInterests}) == 'Sports'">
     <H4>Sport News</H4>
     <esi:include src="/SportNews.jsp?sessionID=$(QUERY_STRING{sessionID})" />
   </esi:when>

   <esi:when test="$(Profile{PersonalInterests}) == 'Career'">
     <H4>Financial News</H4>
     <esi:include src="/FinancialNews.jsp?sessionID=$(QUERY_STRING{sessionID})"
/>
   </esi:when>

   <esi:otherwise>
     <H4>General News</H4>
     <esi:include src="/DefaultNews.jsp?sessionID=$(QUERY_STRING{sessionID})" />
   </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
to disable ESI on the fly without redeveloping or redeploying a different home
page.
</esi:remove>
...
</BODY>
</HTML>
```

Figure 2–9 shows the XML response of `GetProfile.jsp`, which provides access to profile environment variables.

Figure 2–9 GetProfile.jsp XML Response

```
<?xml version=1.0?>
<esi-environment esiversion="ORAESI/9.0.3">
  <PersonalInterests>Sports</PersonalInterests>
  <zipcode>94065</zipcode>
  <tickers>ORCL,YHOO</tickers>
```

```
  <address>500 Oracle Parkway, Redwood Shores, CA 94065</address>
</esi-environment>
```

Fragmentation with the Inline and Include Tags

The <esi:inline> and <esi:include> tags enable applications to adopt ESI page fragmentation and assembly. The following sections describe the tags and explain when the tags are appropriate to use.

- Using Inline for Non-Fetchable Fragmentation
- Using Inline for Fetchable Fragmentation
- Using Include for Fragmentation

Using Inline for Non-Fetchable Fragmentation

Most existing applications are only designed to output an entire Web page to HTTP requests. These fragments and templates are non-fetchable, meaning they are not to be fetched independently from the origin server. If a cache needs any of these fragments or templates, the corresponding full Web page must be requested. To use ESI page assembly for non-fetchable fragments, an application can output the full page response just as it does normally, with the exception that at the beginning and the end of each fragment, an <esi:inline> tag is inserted with a fragment name to demarcate the fragment. Oracle9*i*AS Web Cache stores the enclosed portions as separate fragments and the original page as page templates without the enclosed fragments. Fragments are shared if their names are identical.

Figure 2–10 shows a simple <esi:inline> example. The HTML table enclosed by the <esi:inline> tag is the fragment content. The area preceding <esi:inline name="/news101"> and the area following </esi:inline> form the page template. If another page contains an <esi:inline> tag with the same name "/news101", the two fragments logically share the same content.

Figure 2–10 Inline Non-Fetchable Example

```
<HTML>
...
<esi:inline name="/news101">
<TABLE>
...
</TABLE>
</esi:inline>
...
</HTML>
```

When an application uses non-fetchable <esi:inline> fragments, the full page must be requested for every cache miss. At first, it can appear that there is no apparent cache benefit for cache misses. However, non-fetchable <esi:inline> fragments improves overall caching by:

- Increasing the cache hit ratio

  Because shared fragments can be extracted into separate fragments, the size of the dynamic portion is reduced. A reduced space requirement results in a higher cache hit ratio than full page caching.

- Reducing cache update frequency

  Dynamic shared fragments require only one update. For example, a shared stock market fragment may expire much more frequently than any other parts of the page. With <esi:inline> fragmentation, only one cache update of any full page containing this fragment is enough to bring all full pages sharing this fragment current. Therefore, even non-fetchable <esi:inline> fragments can significantly reduce cache update frequency. The cost reduction is proportional to the degree of sharing.

Using Inline for Fetchable Fragmentation

ESI <esi:inline> fragments are by default non-fetchable. If an application supports independently fetchable fragments, it is possible to use the <esi:inline> for fetchable fragments by setting the fetchable attribute to yes.

Figure 2–11 shows an <esi:inline> example with a fetchable fragment named /news101. A request for the page returns the template page and the fetchable fragment.

Figure 2–11 Inline Fetchable Example

```
<HTML>
...
<esi:inline name="/news101" fetchable="yes">
<TABLE>
...
</TABLE>
</esi:inline>
...
</HTML>
```

> **See Also:** "ESI inline Tag" on page D-5 for further information about the fetchable attribute

Using Include for Fragmentation

The <esi:include> tag is another way to define fragments and templates in an HTTP output for dynamic content caching and assembly. It is in many ways similar to <esi:inline> tag. It defines a name for the defined fragment. The page including an <esi:include> tag is a template that references the defined fragment. However, it also has some key differences which makes its applicable scenarios very different from those of <esi:inline>:

- An <esi:include> tag in a template only defines the reference to a fragment.

  It does not enclose an embedded fragment directly in the template. As a result, a template with <esi:include> tags can be applied to multiple users. In contrast, a template with embedded <esi:inline> tags must be unique to each user.

- A fragment referenced by an <esi:include> tag must always be independently fetchable by HTTP or HTTPS.

  The requested URL is the same as the fragment name. In contrast, an <esi:inline> tag's name only identifies the uniqueness of the fragment and is not used to fetch the actual content. The attribute defining the fragment name in <esi:include> fragment is src instead of name.

There are at least two scenarios where using <esi:include> tags is beneficial:

- Some applications, such as a Web portal, naturally assemble content from external sources. The application only provides a template that is used to fetch various fragments from third-party sources. In this case, the <esi:include> tags fetch and assemble directly, reducing one layer of redundancy.
- Some applications offer faster responses for template-only requests than full-page requests that use <esi:inline> tags. If <esi:include> is used for page fragmentation and assembly, Oracle9*i*AS Web Cache can miss only on the templates when most or all fragments are already cached, saving effective cache miss cost. In many cases, it is also valuable to cache the personalized templates because these seldom change.

Figure 2–8 on page 2-29 shows ESI markup with <esi:include> tags.

Cookie Management for Template Pages and Fragments

Session cookie establishment for ESI templates and fragments works much the same way as typical Oracle9*i*AS Web Cache documents with the following additional features:

- `Cookie` request-header field inheritance

  When a browser requests an ESI template page that includes fragments, requests for fragment pages are generated in Oracle9*i*AS Web Cache. A fragment requests inherit the `Cookie` request-header field from the template request if the value of the `Host` request-header field matches the value of `Host` request-header field in the template request.

- `Set-Cookie` response-header field accumulation

  When assembly of fragments is complete, Oracle9*i*AS Web Cache includes a `Set-Cookie` response-header field in the response with the cookie information from the template. For those fragments with a `Host` request-header field that matches the `Host` request-header field in the template, Oracle9*i*AS Web Cache also accumulates the `Set-Cookie` response-header fields with that of the template. For those fragments with a `Host` request-header field that does match the `Host` request-header field in the template, Oracle9*i*AS Web Cache does not accumulate the `Set-Cookie` response-header field with that of the template and other matching fragments.

See Also:

- "Session Information" on page 2-21 for an overview of `Cookie` and `Set-Cookie` behavior
- "Variable Expressions" on page D-5 for a description of how you can use the `HTTP_COOKIE` variable in ESI markup

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 HTTP or HTTPS fragments of dynamic content, retrieved from the network, into aggregate pages to output to the user. Each fragment can have its own caching rules.

- Support of variables

  ESI supports the use of variables based on HTTP request attributes, as well as custom variables from included HTML fragments. 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 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.

- Character set conversion

  ESI fragments in different character sets are converted to one character set. This way, all partial pages are assembled in a fixed character set. Character set conversion works in the following manner.

  1. Oracle9*i*AS Web Cache receives a request for a template page.
  2. Oracle9*i*AS Web Cache fetches the fragments, and converts all of the fragments to the template's character set. The default character set is ISO-8859-1.

  Oracle9*i*AS Web Cache does not perform character set conversion for non-ESI pages.

- XML conversion to HTML

  Oracle9*i*AS Web Cache uses XSL Transformations (XSLT) to transform XML fragments into HTML.

ESI for Java (JESI)

OC4J provides the JESI tag library as a convenient interface to ESI tags and functionality. Developers have the option of using ESI tags directly in any Web application, but JESI tags provide additional convenience in a JSP environment.

Because ESI and JESI are open standards, you can use the JESI tag library in any standard JSP environment as long as an ESI processor, such as Oracle9*i*AS Web Cache, is available.

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.

Note: The Oracle proprietary language elements described in Table D–2, " Oracle Language Elements" on page D-3 are not supported by the JESI in this release. Oracle Corporation plans to support these language elements in a future implementation.

See Also:

- "Configuring Pages for Content Assembly and Partial Page Caching" on page 7-43
- Appendix D and `http://www.edge-delivery.org` for an overview of the ESI language
- *Oracle9iAS Containers for J2EE JSP Tag Libraries and Utilities Reference* for a description of JESI

Request and Response-Header Fields

Oracle9*i*AS Web Cache provides support for following HTTP request and response-header fields:

- Surrogate-Capability Request-Header Field
- Server Response-Header Field
- Surrogate-Control Response-Header Field

Surrogate-Capability Request-Header Field

For each requested document from the cache, Oracle9*i*AS Web Cache appends a `Surrogate-Capability` request-header field to a document's HTTP request message. The `Surrogate-Capability` request-header enables Oracle9*i*AS Web Cache to identify the operations it is capable of performing to origin server. The `Surrogate-Capability` request-header field supports the following syntax:

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

where "*`operation_value`*" is one or more of the following:

- `"ORAESI/9.0.2"` to process ESI tags with Oracle proprietary additions for content assembly and partial page caching. `"ORAESI/9.0.2"` supports all the ESI tags provided by Oracle9*i*AS Web Cache in release 9.0.2 or 9.0.3.
- `"ESI/1.0"` to process standard ESI tags for content assembly and partial page caching
- `"ESI-Inline/1.0"` to process `<esi:inline>` tags
- `"webcache/1.0"` to process the `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags for personalized attributes and session-encoded URLs

`"webcache/1.0"` is mutually exclusive with `"ORAESI/9.0.2"`,`"ESI/1.0"`, and `"ESI-Inline/1.0"`.

> **See Also:** "Configuring Caching Attributes in Response Headers" on page 7-66

Server Response-Header Field

For documents sent to browsers, Oracle9*i*AS Web Cache adds diagnostic information to the `Server` response-header field of the HTTP response message:

`Server: Oracle9iAS/`*`version server_header_from_origin_server`*
`Oracle9iAS-Web-Cache/`*`version`* `(`*`diagnostic_information`*`)`

The `Server` response-header field specifies name/value pairs for Oracle HTTP Server and Oracle9*i*AS Web Cache. The information for Oracle9*i*AS Web Cache includes version and diagnostic information.

`diagnostic_information` has the following format:

`{`*`ESI_processing_type`*`}{`*`cache_request_type`*`}[;max-age=`*`expiration_time`*`[+`*`removal_time`*`];age=`*`document_age`*`]`

Table 2–4 describes the diagnostic fields.

Table 2–4 Control Directives for Surrogate-Control

Control Directive	Description
`ESI_processing_type`	*`ESI_processing_type`* can be one of the following: ■ `T` specifies that the document is an ESI template ■ `F` specifies that the document is an ESI fragment ■ empty specifies that the response does not require ESI processing
`cache_request_type`	*`cache_request_type`* can be one of the following: ■ `H` specifies a cache hit ■ `S` specifies a cache hit of a stale document ■ `U` specifies a cache update of a stale document ■ `G` specifies a cache update of a document that was requested to removed but still physically resides in the cache ■ `M` specifies a cacheable cache miss ■ `N` specifies a non-cacheable cache miss
`max-age="`*`expiration_time`*`[+ `*`removal_time`*`]`	Specifies the time, in seconds, to expire the document, and optionally, the time, in seconds, to remove the document from the cache after the expiration time. `max_age` does not appear if the *`cache_request_type`* is `N`.
`age=`*`document_age`*	Shows how long, in seconds, the document has been in the cache. `age` does not appear if the document is non-cacheable.

Using the `Server` response-header information, you can determine whether a request was served from the cache or the application Web server. In the following example, the `Server` field specifies that the document was a cache hit:

```
Server: Oracle9iAS/9.0.3 Oracle HTTP Server Powered by Apache/1.3.12 (Unix)
mod_plsql/3.0.9.8.3b ApacheJServ/1.1 mod_perl/1.24
Oracle9iAS-Web-Cache/9.0.3.0.0 (TH;max-age=60+30;age=55)
```

`(TH;max-age=60+30;age=55)` is the diagnostic information.

- `T` means this page is composed by ESI
- `H` means this request resulted in cache hit
- `max-age=60+30` means that the document is to expire in 60 seconds from population and to be removed from the cache 30 seconds from the expiration. This provides a total of 90 seconds from population.
- `age=55` in age means that 55 seconds have passed since population of the cache, meaning there is 5 seconds to expiration and 35 seconds to removal

> **See Also:** "Diagnostic Information in the Server Response-Header Field or HTML Body" on page 10-9 for instructions on disabling the diagnostics information or displaying the information in the HTML response

Surrogate-Control Response-Header Field

Application developers can choose to store some of the caching attributes in a `Surrogate-Control` response-header field header. This response-header field enables the application Web server to override the settings configured through the Oracle9*i*AS Web Cache Manager interface. It is also required to enable partial page caching for pages supporting ESI.

The `Surrogate-Control` response-header field supports the following syntax:

```
Surrogate-Control:[content=content_type, content_type,..]
[no-store][no-store-remote][max-age=expiration_time[+ removal_
time]][vary=headers(header header...); coookie(cookie_name cookie_name...)]
```

> **See Also:** "Configuring Caching Attributes in Response Headers" on page 7-66 for complete information about configuring the supported `Surrogate-Control` response-header field

3

Cache Clustering

You can configure multiple instances of Oracle9*i*AS Web Cache to run as independent caches, with no interaction with one another. Most of the deployment scenarios in this guide describe this type of configuration.

However, to increase the availability and scalability of your Web cache, you can configure multiple instances of Oracle9*i*AS Web Cache to run as members of a cache cluster. A **cache cluster** is a loosely coupled collection of cooperating Web cache instances working together to provide a single logical cache.

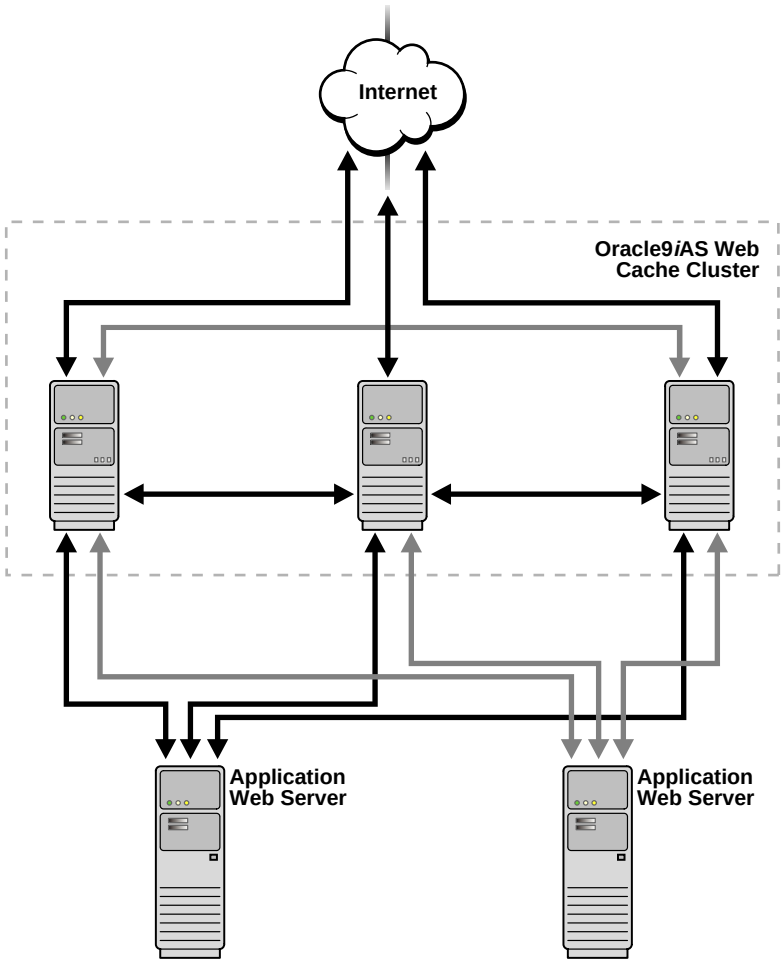
This chapter contains the following topics:

Overview of Cache Clusters

In a cache cluster, multiple instances of Oracle9*i*AS Web Cache, the **cache cluster member**, operate as one logical cache. A cache cluster can consist of two or more members. The cache cluster members communicate with one another to request cacheable content that is cached by another cache cluster member and to detect when a cache cluster member fails.

Figure 3–1 shows an Oracle9*i*AS Web Cache cluster that contains three cache cluster members. As the figure shows, the cluster members communicate with one another as well as with the application Web servers and with the clients.

Figure 3–1 Oracle9iAS Web Cache Cluster Architecture



Oracle9*i*AS Web Cache uses the relative capacity of each cache instance to distribute the cached content among the cache cluster members. In effect, it assigns a cache cluster member to be the owner of a particular document. This content is called **owned content.**

In addition to the owned content, Oracle9*i*AS Web Cache stores popular documents in the cache of each cluster member. These documents are known as **on-demand content**. By storing the on-demand content, Oracle9*i*AS Web Cache responds to requests for those documents quickly and decreases the number of cache misses.

Fewer requests are sent to the application Web server. The result is improved performance.

Benefits of Cache Clusters

Cache clusters provide the following benefits:

- High availability

  With or without cache clusters, Oracle9*i*AS Web Cache ensures that cache misses are directed to the most available, highest-performing Web server. With cache clusters, Oracle9*i*AS Web Cache supports failure detection and failover of Web caches. If a Web cache fails, other members of the cache cluster detect the failure and take over ownership of the cacheable content of the failed cluster member.

- Scalability and performance

  By distributing the site's content across multiple Web caches, more content can be cached and more client connections can be supported, expanding the capacity of your Web site.

  By deploying multiples caches in a cache cluster, you make use of the processing power of more CPUs. Because multiple requests are executed in parallel, you increase the number of requests that are served concurrently.

  Network bottlenecks often limit the number of requests that can be processed at one time. Even on a node with multiple network cards, you can encounter operating system limitations. By deploying caches on separate nodes, more network bandwidth is available. Response time is improved because of the distribution of requests.

  In a cache cluster, fewer requests are routed to the application Web server. Retrieving content from a cache (even if that request is routed to another cache in the cluster) is more efficient than materializing the content from the application Web server.

- Reduced load on the application Web server

  In a cache cluster environment, popular documents are stored in more than one cache. If a cache fails, requested cacheable documents are likely to be stored in in the cache of surviving cluster members. As a result, fewer requests for cacheable documents need to be routed to the application Web server even when a cache fails.

When a failed cache returns to operation, it has no documents cached. In a noncluster environment with multiple independent caches, that cache must route cache misses to the application Web server. In a cache cluster environment, that cache can route cache misses to other caches in the cluster, reducing the load on the application Web server.

Cache clusters maximize system resource utilization. When each cache in a cache cluster resides on a separate node, more memory is available than for one cache on a single node. With more memory, Oracle9*i*AS Web Cache can cache more content, resulting in fewer requests to the application Web server.

- Improved data consistency

  Because Oracle9*i*AS Web Cache uses one set of invalidation rules for all cache cluster members and because it makes it easy to propagate invalidation requests to all cache cluster members, the cached data is more likely to be consistent across all caches in a cluster.

- Manageability

  Cache clusters are easy to manage because they use one configuration for all cache cluster members. For example, you specify one set of cacheability rules and one set of invalidation rules. Oracle9*i*AS Web Cache copies those rules to all cluster members by propagating the configuration to all cluster members.

How Cache Clusters Work

In a cache cluster, multiple instances of Oracle9*i*AS Web Cache operate as one logical cache.

A cache cluster uses one configuration that is propagated to all cluster members. The configuration contains general information, such as security, session information, and cacheability rules, which is the same for all cluster members. It also contains cache-specific information, such as capacity, administration and other ports, resource limits, and log files, for each cluster member.

Each member must be authenticated before it is added to the cache cluster. The authentication requires that the administration username and password of the Oracle9*i*AS Web Cache instance to be added be the same as the administration username and password of the cluster.

When you add a cache to the cluster, the cache-specific information of the new cluster member is added to the configuration of the cache cluster. Then, Oracle9*i*AS Web Cache propagates the configuration to all members of the cluster. Because adding a new member changes the relative capacity of each Web cache, Oracle9*i*AS

Web Cache uses the information about capacity to recalculate which cluster member owns which content.

When cache cluster members detect the failure of another cluster member, the remaining cache cluster members automatically take over ownership of the content of the failing member. When the cache cluster member is reachable again, Oracle9*i*AS Web Cache again reassigns the ownership of the content.

When you remove a Web cache from a cache cluster, the remaining cache cluster members take over ownership of the content of the removed member. In addition, the configuration information about the removed member is deleted from the configuration and the revised configuration is propagated to the remaining cache cluster members.

In a cache cluster, administrators can decide whether to propagate invalidation messages to all cache cluster members or to send invalidation messages individually to cache cluster members.

> **See Also:** "Invalidation Propagation" on page 2-7 for more information about invalidation propagation in cache clusters

How Cache Content Is Distributed

Oracle9*i*AS Web Cache uses the relative capacity of each cache to automatically distribute ownership of documents among the cache cluster members. For example, in a three-cache cluster, if cache_X has a capacity of 10, cache_Y has a capacity of 10, and cache_Z has a capacity of 20, Oracle9*i*AS Web Cache distributes ownership of 25% of the cached content to cache_X, 25% of the cached content to cache_Y, and 50% of the cached content to cache_Z.

Oracle9*i*AS Web Cache maintains a structure to record ownership of documents. When a request for a cacheable document is received, Oracle9*i*AS Web Cache uses the structure to assign a cache cluster member to be the owner of the document.

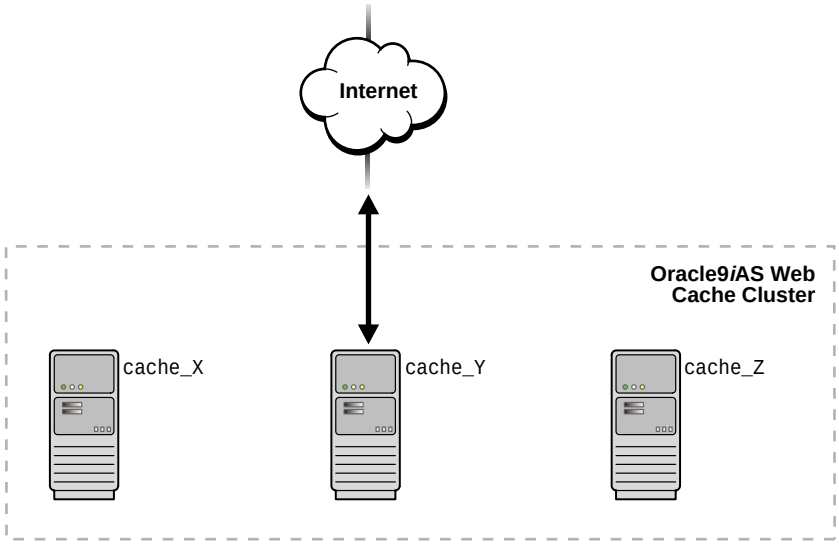
Note that in addition to the owned content, Oracle9*i*AS Web Cache stores popular documents (on-demand content) in the Web cache of each cluster member. By storing the on-demand content, Oracle9*i*AS Web Cache returns future requests for those documents quickly and decreases the number of cache misses. The result is improved performance.

When an incoming request for a noncacheable document is received by one of the cache cluster members, the requested is forwarded to the application Web server.

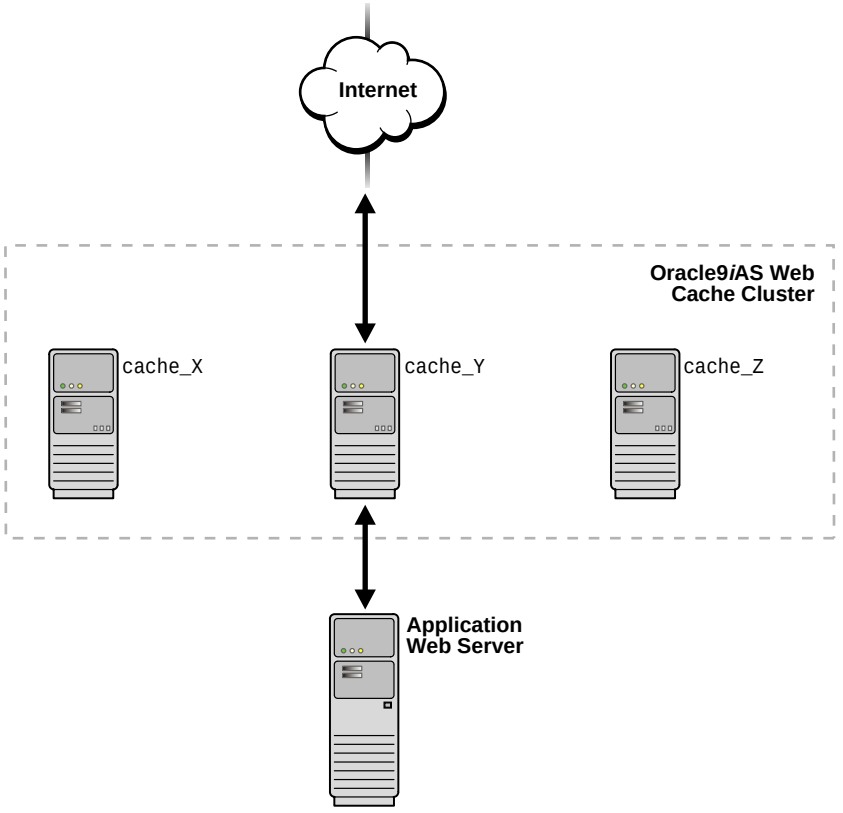
When an incoming request for a cacheable document is received by one of the cache cluster members, what happens next depends on whether or not the requested

content is cached by that cluster member and whether or not the content is owned by that cluster member. Suppose that cluster member cache_Y receives a request for cacheable content:

- If the content is cached by cache_Y, cache_Y returns the content to the client, as shown in the following figure. The content could be either owned content or on-demand content.

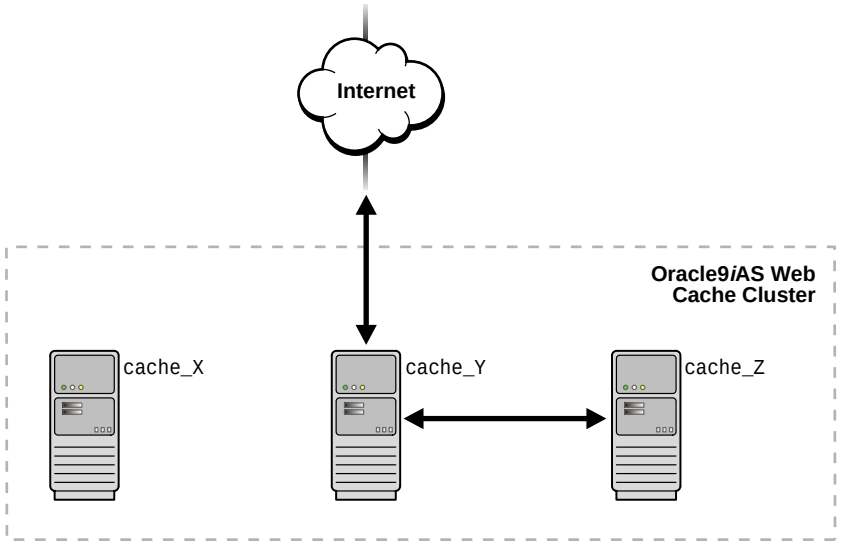


- If the content is not cached by cache_Y, Oracle9*i*AS Web Cache performs an ownership lookup. Then:
  - If cache_Y is the owner of the requested content, it sends the request to the application Web server, which returns the requested content to cache_Y. Then, cache_Y caches the content and returns it to the client, as shown in the following figure:

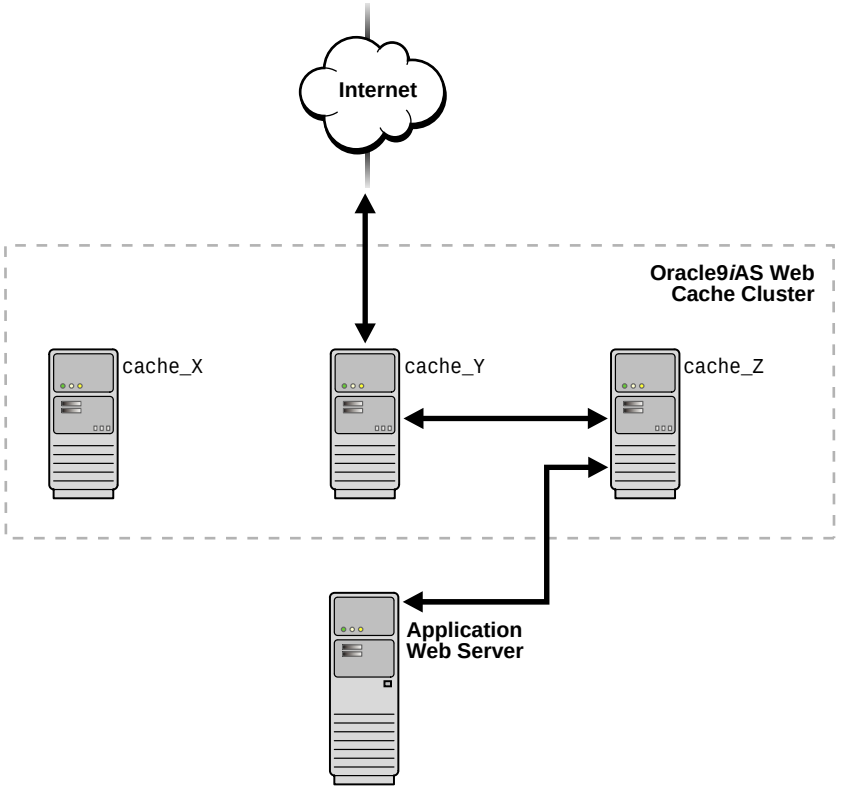


– If cache_Y is the not owner of the requested content, cache_Y sends the request to the owner cache, cache_Z. Then:

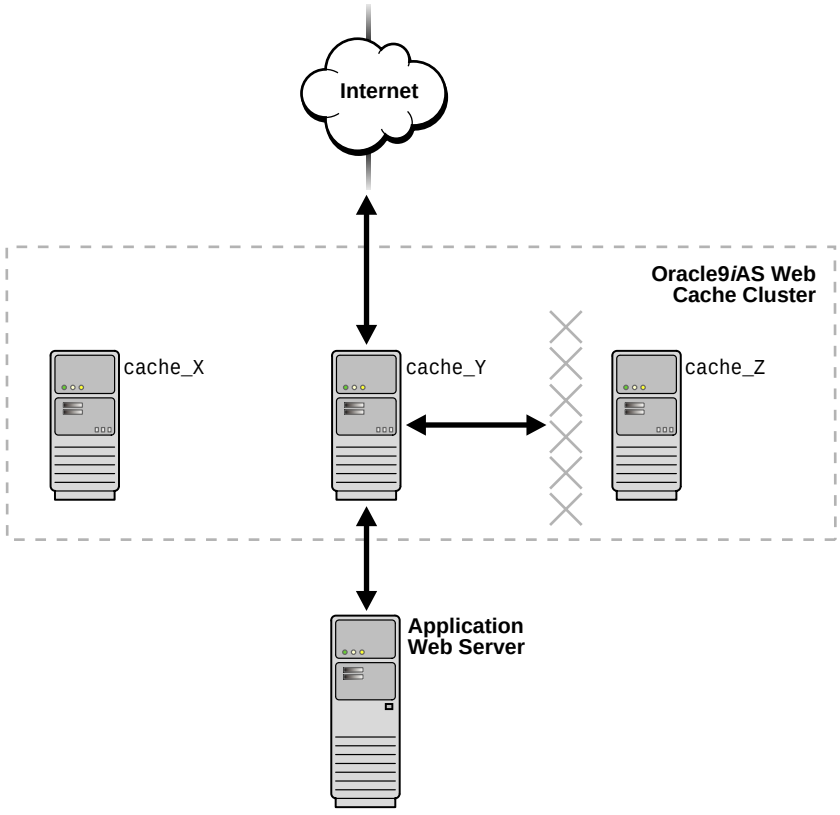
  * If the requested content is stored in the owner's cache, the owner returns the requested content to the cluster member (cache_Y) that originally received the request. Cache_Y returns the requested content to the client. The content is stored in cache_Y as on-demand content to satisfy future requests for that document quickly.



* If the requested content is not stored in the owner's cache, the owner, cache_Z, sends the request to the application Web server. The application Web server returns the requested content to the owner. The owner caches the content and sends the requested content to the cluster member (cache_Y) that originally received the request. Cache_Y returns the requested content to the client and stores the content as on-demand content.



- If cache_Y cannot contact the owner (because of network problems or because the server for the owner has failed), it sends the request to the application Web server. The application Web server returns the requested content to cache_Y. Then, cache_Y returns the requested content to the client and stores the content as on-demand content.



When you add a member to or remove a member from the cache cluster, Oracle9*i*AS Web Cache uses the information about capacity to recalculate which cluster member owns which documents. If the ownership of a document changes and the document is currently cached, Oracle9*i*AS Web Cache designates the document as on-demand content, rather than owned content, for that cache. The document is not removed from the previous owner cache, nor is it moved to the new owner cache. The document is not cached in the new owner cache until another request for the document is received.

Failure Detection and Failover

Oracle9*i*AS Web Cache clusters ensure high availability through failure detection and failover. In clusters, **failure detection** ensures that Oracle9*i*AS Web Cache can detect when a cache cluster member is unavailable; **failover** ensures that Oracle9*i*AS Web Cache transfers ownership of the content of the failing member to the remaining cluster members.

Cache cluster members send requests to the cluster member who is the owner of the requested content. If a cache cluster member does not receive a response from another cluster member after a specified failover threshold (the number of consecutive attempts to reach a cache), the cache cluster member assumes that the other cluster member has failed. Each cluster member individually detects the failure of other cluster members.

As each cache cluster member detects the failure of another cluster member, it recalculates the relative capacity of the remaining cache cluster members. Then, it reassigns ownership of documents based on the new relative capacity and the ownership array. Note that although ownership is reassigned, the content is not cached in the new owner cache until another request for the document is received.

The cache cluster members poll the failed Web cache server for its current status until it is reachable again. When the failed Web cache is reachable, it rejoins the cache cluster. Each cache cluster member again recalculates the relative capacity of the cache cluster members and reassigns ownership of the documents.

> **See Also:** Configuring a Cache Cluster on page 6-55 for information about configuring a cache cluster, including specifying a failover threshold.

4

Deploying Oracle9*i*AS Web Cache

Note: Many of the deployment scenarios described in this chapter can use a **cache cluster** in place of one Oracle9*i*AS Web Cache server.

This chapter presents several high-level scenarios for deploying Oracle9*i*AS Web Cache.

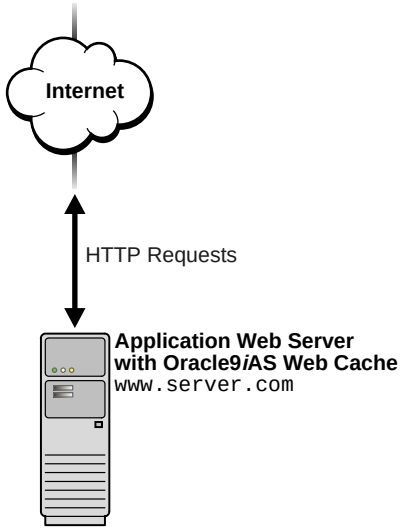
This chapter contains these topics:

Caching Content for One Application Web Server

Oracle9*i*AS Web Cache can be deployed on the same computer as the application Web server or on a separate computer.

Figure 4–1 shows Oracle9*i*AS Web Cache deployed on the same computer as the application Web server.

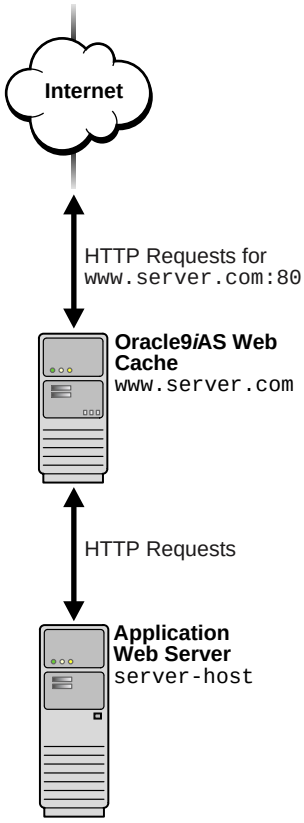
Figure 4–1 Oracle9iAS Web Cache On the Same Computer As the Application Web Server



For this deployment, configure Oracle9*i*AS Web Cache with application Web server settings and `www.server.com` site settings.

For optimal performance, Oracle Corporation recommends deploying Oracle9*i*AS Web Cache on a dedicated, fast two-CPU computers with lots of memory. Figure 4–2 shows Oracle9*i*AS Web Cache deployed on a different computer from the application Web server.

Figure 4–2 Oracle9iAS Web Cache On a Different Computer From the Application Web Server



To configure this deployment:

1. Register the IP address of the Oracle9*i*AS Web Cache server with `www.server.com`.
2. Rename the application Web server, and assign the computer running Oracle9*i*AS Web Cache the name that was previously assigned to the application Web server.

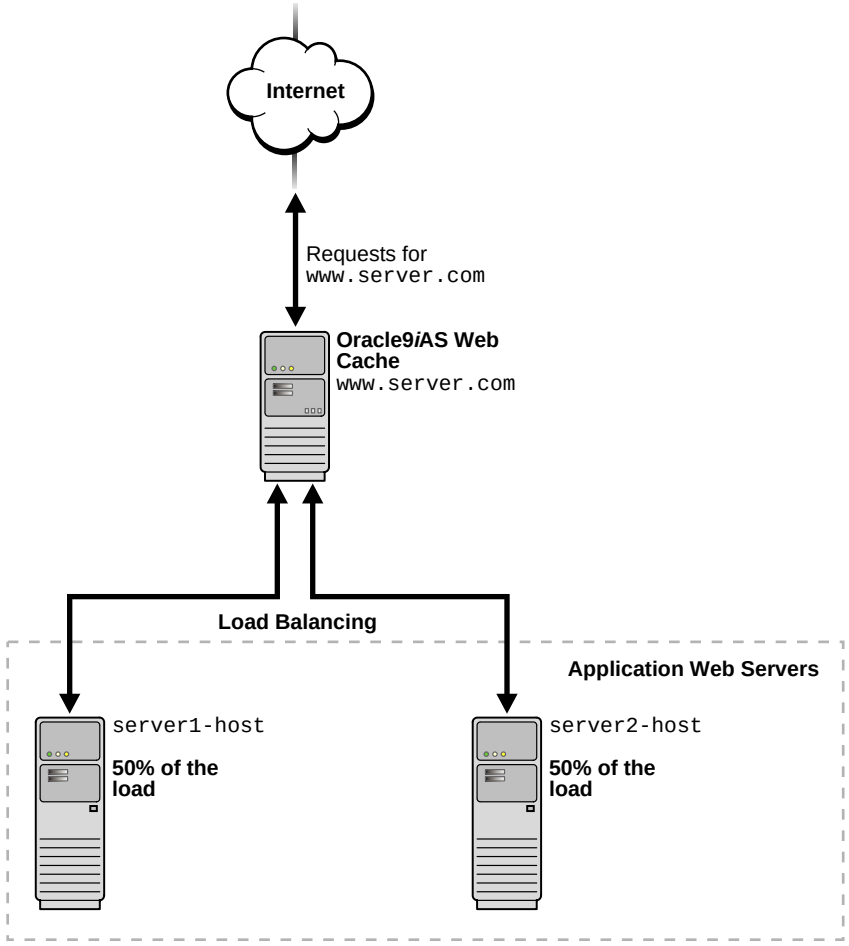
   In Figure 4–2, Oracle9*i*AS Web Cache is named `www.server.com`, which was the name of the application Web server. The application Web server is renamed to `server-host`.
3. Configure the Oracle9*i*AS Web Cache server with the following:
   - Application Web server settings for `server1-host`
   - Site settings for `www.server.com`

Note: In configurations with a **Load Balancer**, register the IP address of the Load Balancer rather than the Oracle9*i*AS Web Cache server with the Web site's domain name. See "Load Balancing Requests Among Application Web Servers" on page 4-5 for more information about deployments with a Load Balancer.

Load Balancing Requests Among Application Web Servers

Many of today's Web sites use a Load Balancer to balance the incoming requests among multiple application Web servers. Instead, as shown in Figure 4–3, you can use Oracle9*i*AS Web Cache to distribute requests among two or more application Web servers.

Figure 4–3 Load Balancing with Oracle9iAS Web Cache



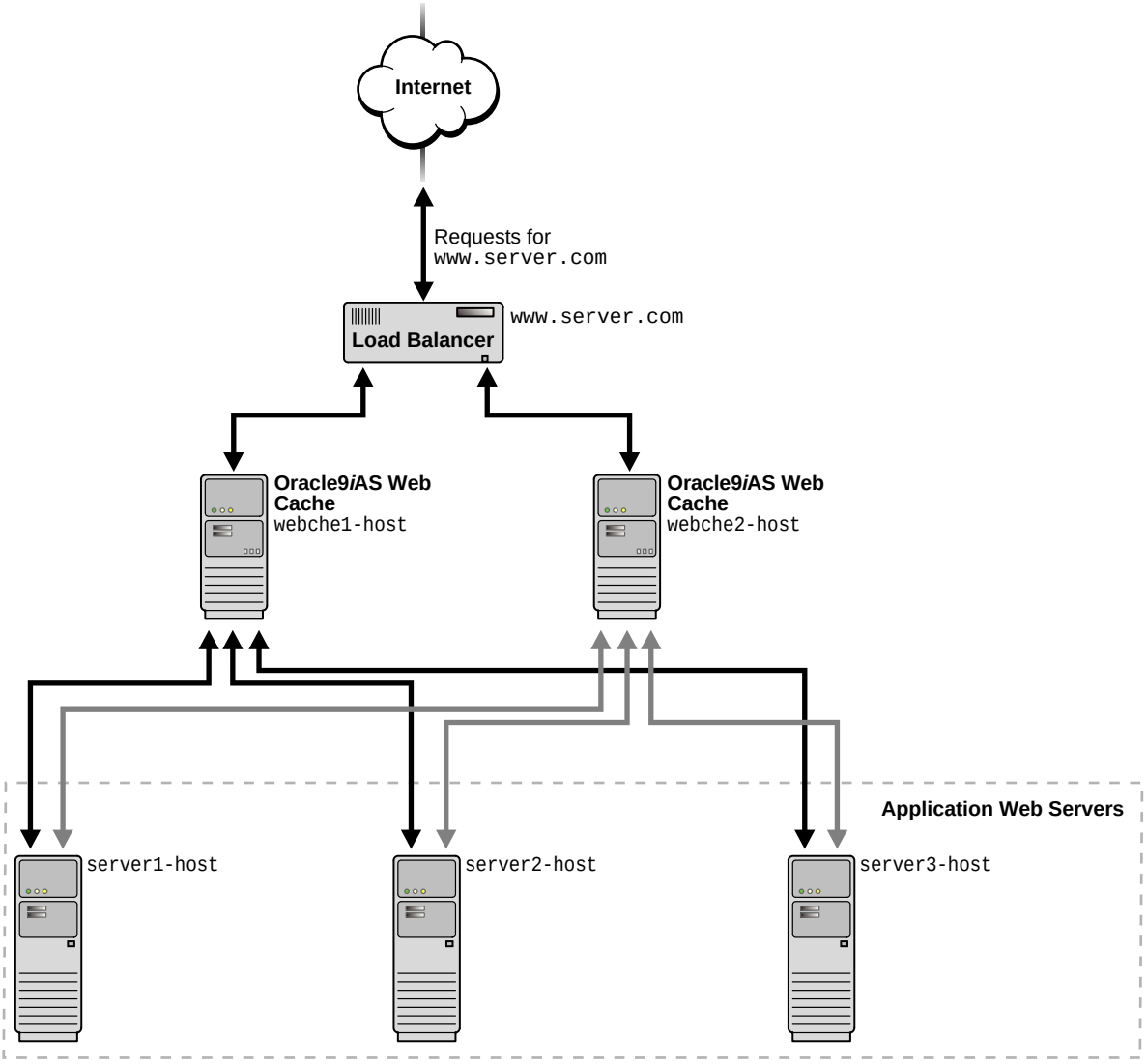
To configure this deployment:

1. Assign the name of the Load Balancer to Oracle9*i*AS Web Cache.
2. Register the IP address of the Oracle9*i*AS Web Cache server with `www.server.com`.
3. Configure the Oracle9*i*AS Web Cache server with the following:
   - Application Web server settings for `server1-host` and `server2-host`
   - Site settings for `www.server.com`

Using Oracle9*i*AS Web Cache Servers in a Failover Pair

To maintain performance during an application Web server failure, you can configure two Oracle9*i*AS Web Cache servers as a failover pair. Both Oracle9*i*AS Web Cache servers are configured to cache the same content. When both Oracle9*i*AS 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 4–4 on page 4-7.

Figure 4–4 Configuring Multiple Oracle9iAS Web Caches as a Failover Pair



To configure this deployment:

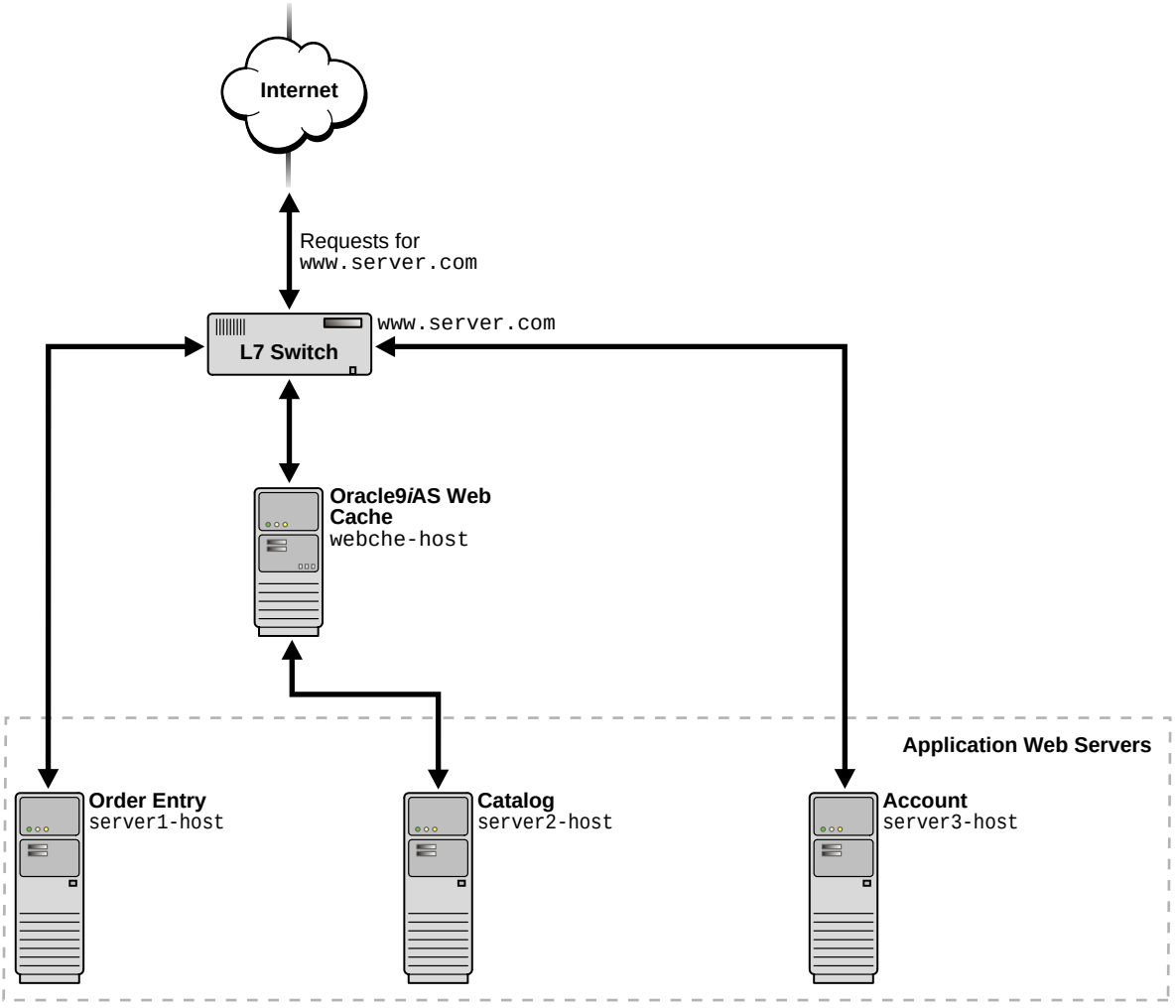
1. Register the IP address of the Load Balancer with `www.server.com`.
2. Configure the Load Balancer with Oracle9*i*AS Web Cache server host names `webche1-host` and `webche2-host`.
3. Configure each of the Oracle9*i*AS Web Cache servers with the following:
   - Application Web server settings for `server1-host`, `server2-host`, and `server2-host`
   - Site settings for `www.server.com`

Accelerating Portions of a Web Site

Many Web sites contain cacheable public content and non-cacheable content. For these Web sites, you can use Oracle9*i*AS Web Cache servers to cache content for just the portions of the Web site with the cacheable content. Figure 4–5 on page 4-9 shows a **Layer 7 (L7) switch** passing catalog requests to Oracle9*i*AS Web Cache server `webche-host` and order entry and account requests to application Web servers `server1-host`, `server2-host`, and `server3-host`. An L7 switch operates at Layer 7, the Application Layer layer, of the **Open Systems Interconnection (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 4–5 Accelerating Portions of a Web Site



To configure this deployment:

1. Register the IP address of the L7 switch with `www.server.com`.
1. Configure the L7 switch with Oracle9*i*AS Web Cache server host name `webche-host`.
2. Configure Oracle9*i*AS Web Cache server `webche-host` with the following:
   - Application Web server settings for `server1-host`, `server2-host`, and `server3-host`
   - Site settings for `www.server.com`

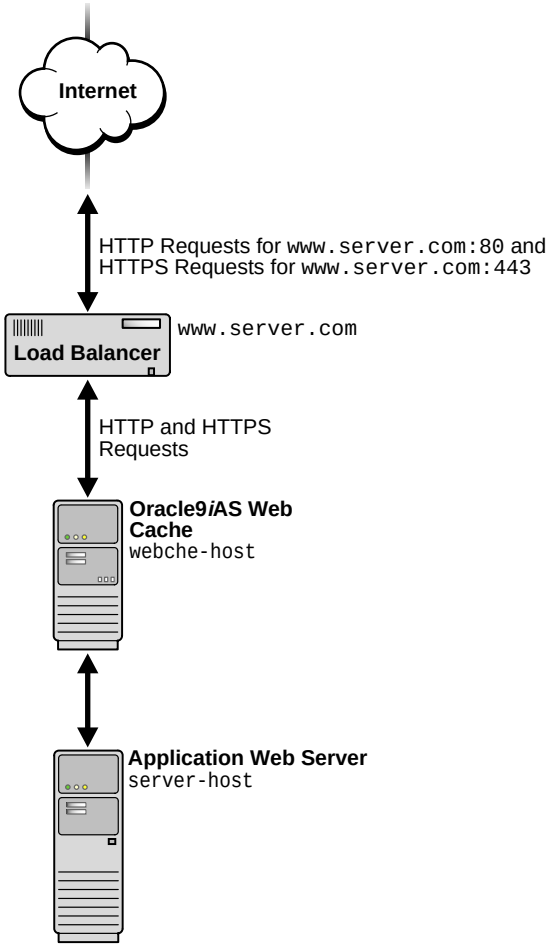
Caching Content for HTTPS Requests

In addition to **HTTP protocol** requests, you can configure Oracle9*i*AS 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 Oracle9*i*AS Web Cache to listen for incoming requests on two ports, one for HTTPS requests and one for HTTP requests. Typically, HTTP uses port 80 and HTTPS uses port 443. A Load Balancer can be configured to pass requests to the appropriate listening port.

You can also configure Oracle9*i*AS Web Cache to send traffic to the application Web server through an HTTP or HTTPS listening port.

Figure 4–6 on page 4-11 shows a Load Balancer passing both HTTP and HTTPS requests to Oracle9*i*AS Web Cache server `webche-host`.

Figure 4–6 Deploying Oracle9iAS Web Cache to Receive HTTP and HTTPS Requests



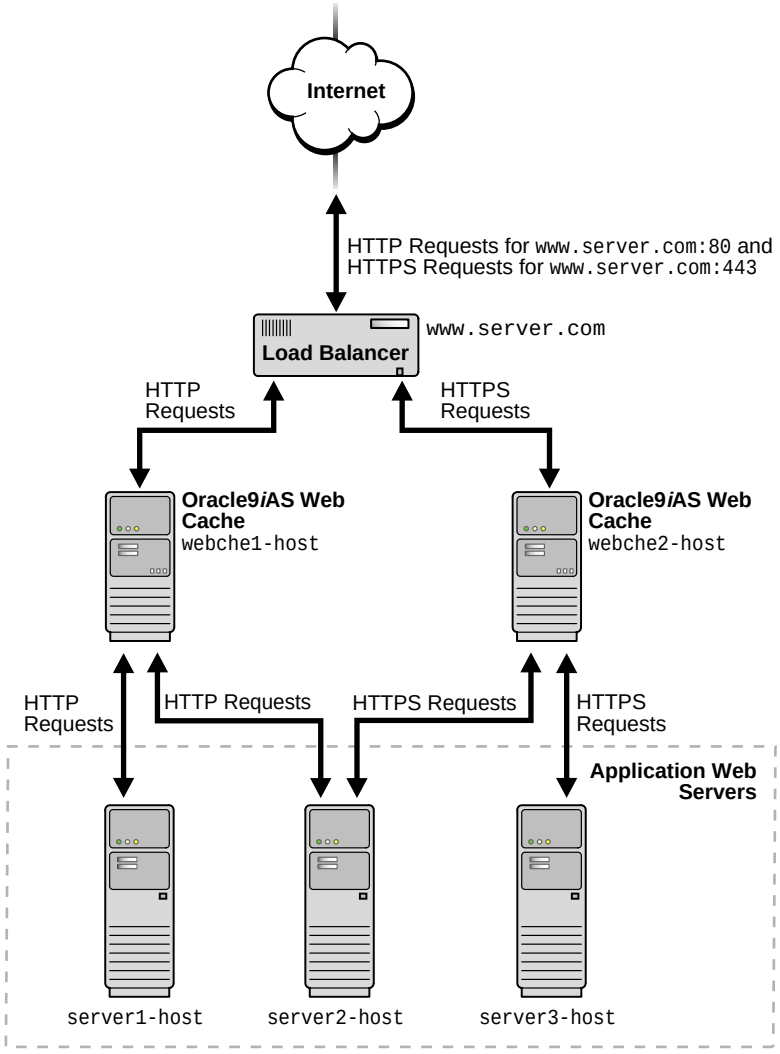
To configure this deployment:

1. Register the IP address of the Load Balancer with `www.server.com`.
2. Configure the Load Balancer with Oracle9*i*AS Web Cache server host name `webche-host`.

3. Configure Oracle9*i*AS Web Cache server `webche-host` with the following:
   - Receive requests on HTTP and HTTPS listening ports
   - Send requests to application Web server `server-host` on an HTTP or HTTPS listening port
   - Site settings for `www.server.com:80` and `www.server.com:443`

Figure 4–7 on page 4-13 shows two Oracle9*i*AS Web Cache servers receiving requests. HTTP requests are served from server `webche1-host` and HTTPS requests are served from server `webche2-host`. Oracle9*i*AS Web Cache server `webche1-host` sends HTTP requests to application Web servers `server1-host` and `server2-host`. Oracle9*i*AS Web Cache server `webche2-host` sends HTTPS requests to application Web servers `server2-host` and `server3-host`.

Figure 4–7 Forwarding HTTPS Requests To a Dedicated Oracle9iAS Web Cache Server

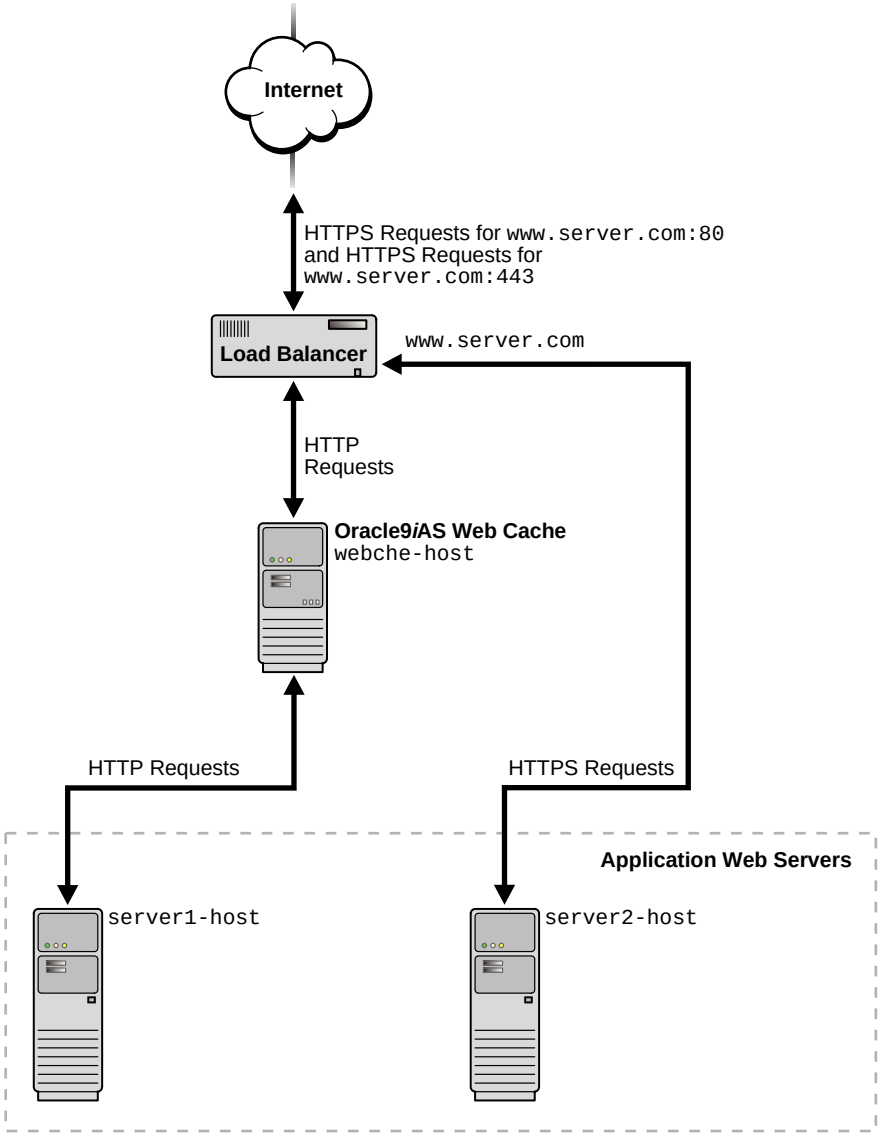


To configure this deployment:

1. Register the IP address of the Load Balancer with `www.server.com`.
2. Configure the Load Balancer with Oracle9*i*AS Web Cache server host names `webche1-host` and `webche2-host`.
3. Configure Oracle9*i*AS Web Cache server `webche1-host` with the following:
   - Receive requests on an HTTP listening port
   - Send requests to application Web servers `server1-host` and `server2-host` on HTTP listening ports
   - Site settings for `www.server.com:80`
4. Configure Oracle9*i*AS Web Cache server `webche2-host` with the following:
   - Receive requests on an HTTPS listening port
   - Send requests to application Web servers `server2-host` and `server3-host` on HTTPS listening ports
   - Site settings for `www.server.com:443`

For many applications, HTTPS is required for secure transactions that should not 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 a Load Balancer to pass all HTTP requests to Oracle9*i*AS Web Cache, and forward HTTPS requests for secure pages to a particular application Web server. Figure 4–8 on page 4-15 shows a Load Balancer passing HTTP requests to Oracle9*i*AS Web Cache server `webche-host` and HTTPS requests to application Web server `server2-host`. Note that HTTPS requests could also be passed to `server1-host`.

Figure 4–8 Forwarding HTTPS Requests To an Application Web Server



Using Oracle9*i*AS Web Cache to Support Multiple Sites

To cache content for multiple internal or external Web sites, you can configure Oracle9*i*AS Web Cache to cache content for a **virtual host site**, and cache and assemble HTML fragments for **Edge Side Includes (ESI)** `<esi:include>` requests from an **ESI provider site**.

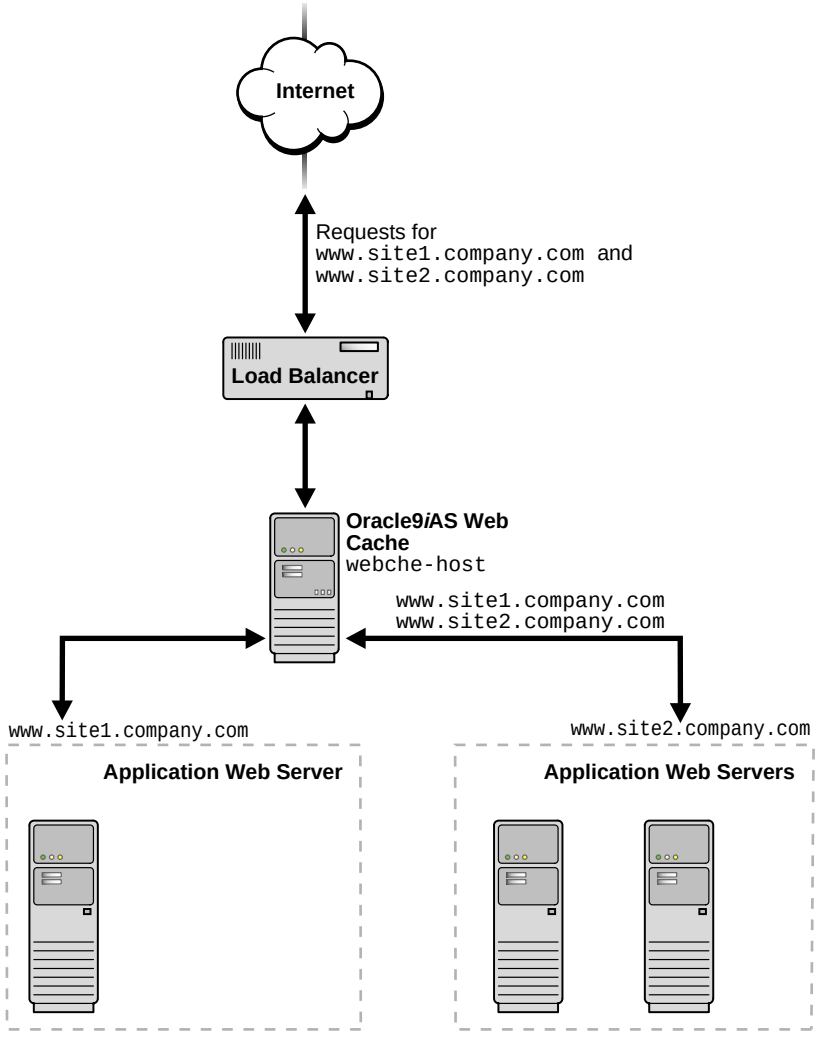
This section depicts the following deployments:

- Multiple Internal Virtual Host Sites
- Multiple Internal ESI Provider Sites
- Multiple External Sites

Multiple Internal Virtual Host Sites

Figure 4–9 on page 4-17 shows an internal virtual host deployment. It shows Oracle9*i*AS Web Cache server `webche-host` serving content on behalf of internal virtual host sites `www.site1.company.com` and `www.site2.company.com`.

Figure 4–9 Configuring Support for Multiple Internal Virtual Host Sites



To configure this deployment:

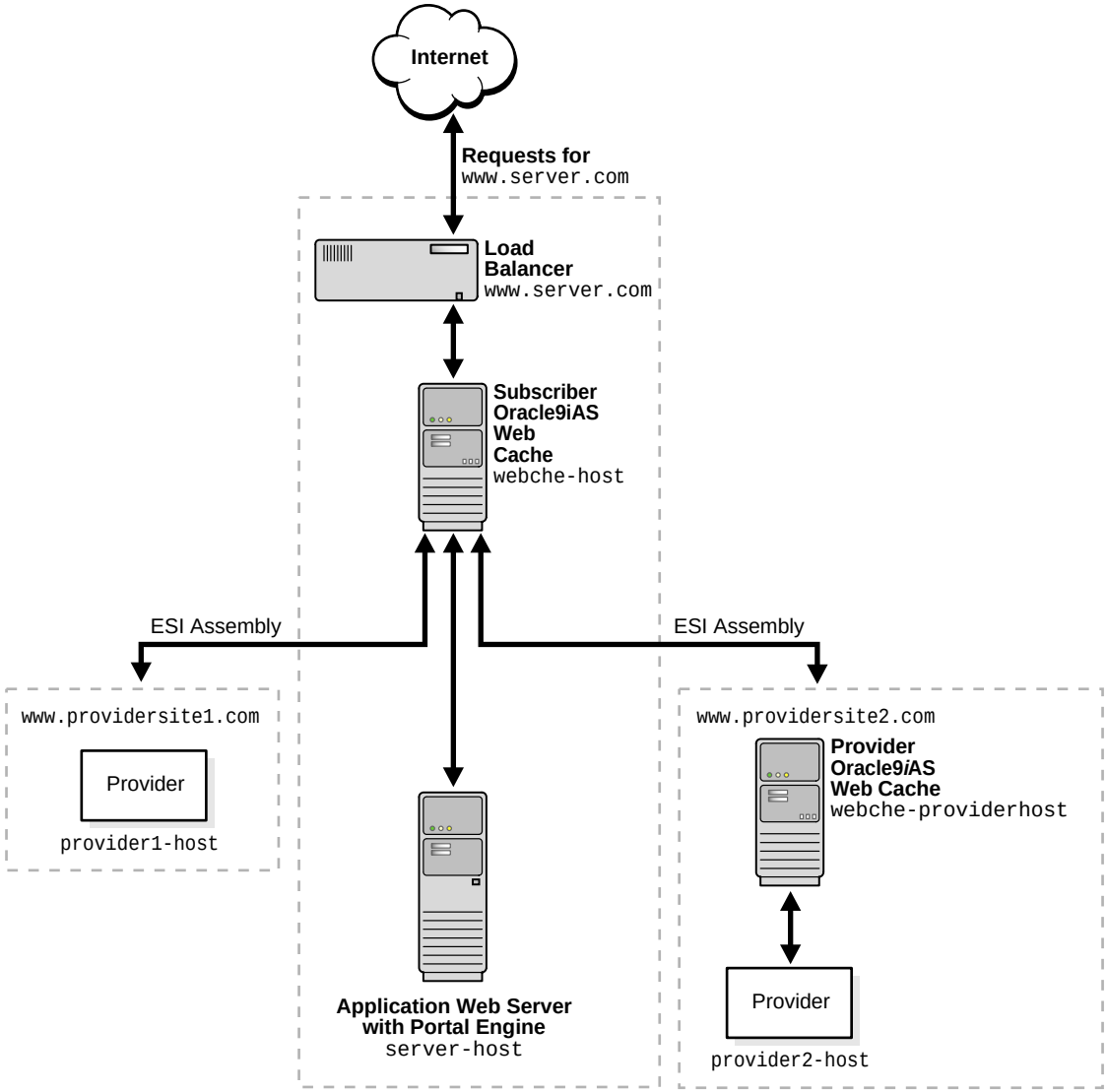
1. Register the IP address of the Load Balancer with `www.site1.company.com` and `www.site2.company.com`.
2. Configure the Load Balancer with Oracle9*i*AS Web Cache server host name `webche-host`.

3. Configure Oracle9*i*AS Web Cache server `webche-host` with the following:
   - Virtual host site definitions for `www.site1.company.com` and `www.site2.company.com`
   - Map the site definitions to the appropriate application Web servers

Multiple Internal ESI Provider Sites

Figure 4–10 on page 4-19 shows an internal ESI provider site deployment. It shows Oracle9*i*AS Web Cache server `webche-host` assembling ESI content from internal ESI provider sites `www.providersite1.com` and `www.providersite2.com`. Application Web server `server-host` uses a portal application to create a template page and sends it back to `webche-host` for assembly. `webche-host` includes ESI fragments for the template page from `www.providersite1.com` and Oracle9*i*AS Web Cache server `webcache-providerhost`, which is caching content for `www.providersite2.com`.

Figure 4–10 Configuring Support for Multiple Internal ESI Provider Sites



To configure this deployment:

1. Register the IP address of the Load Balancer with `www.server.com`.
2. Configure the Load Balancer with Oracle9*i*AS Web Cache server host name `webche-host`.
3. Configure Oracle9*i*AS Web Cache server `webche-host` with the following:
   - `server-host`, `provider1-host`, and `webche-providerhost` as the application Web servers
   - `www.server.com` as a virtual host site mapped to `server-host`
   - `www.providersite1.com` as an ESI provider site mapped to `provider1-host`
   - `www.providersite2.com` as an ESI provider site mapped to `webche-providerhost`
4. Configure Oracle9*i*AS Web Cache server `webche-providerhost` with the following.
   - `provider2-host` as the application Web server
   - `www.providersite2.com` as an ESI provider site mapped to `provider2-host`

   See Also: "Configuring an ESI Cache Hierarchy" on page 6-48 for more details about this configuration

Multiple External Sites

Many virtual host sites and ESI provider sites are likely to be connected over the Internet protected by a firewall and accessible through a proxy server, as shown in Figure 4–11 and Figure 4–12 on page 4-22. For these types of sites, you configure Oracle9*i*AS Web Cache with proxy server settings rather than application Web server settings.

Figure 4–11 Configuring Support for Multiple External Virtual Host Sites

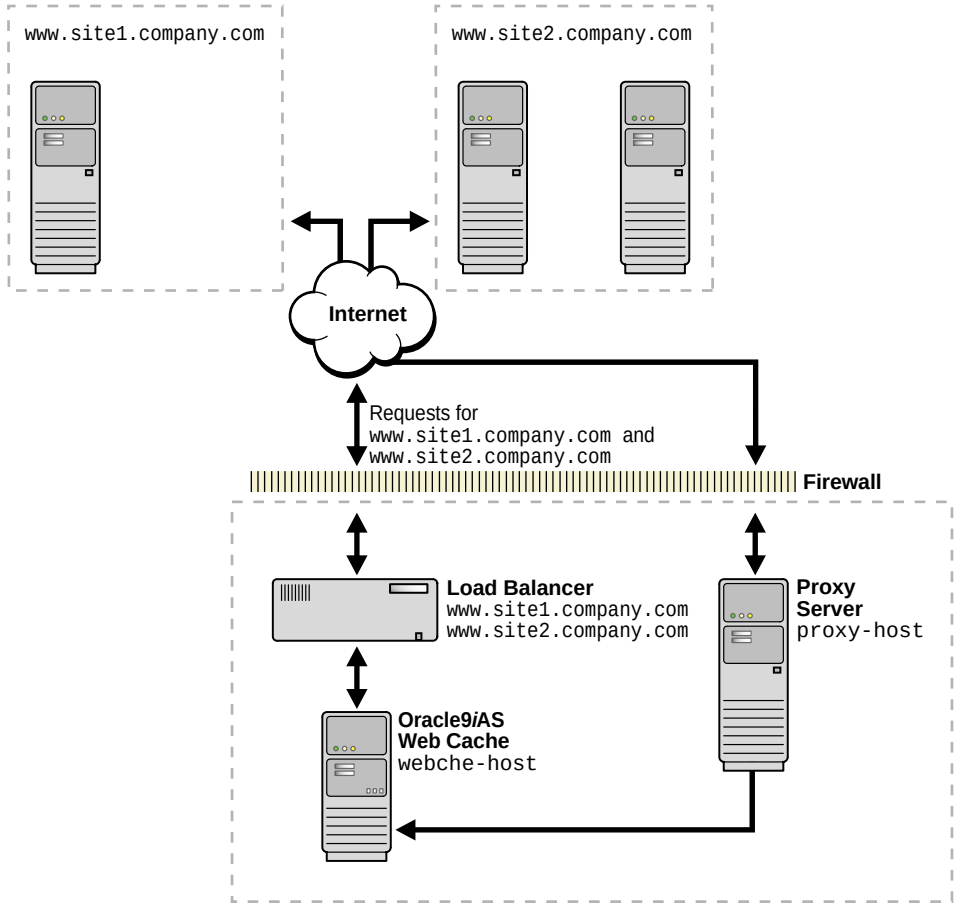
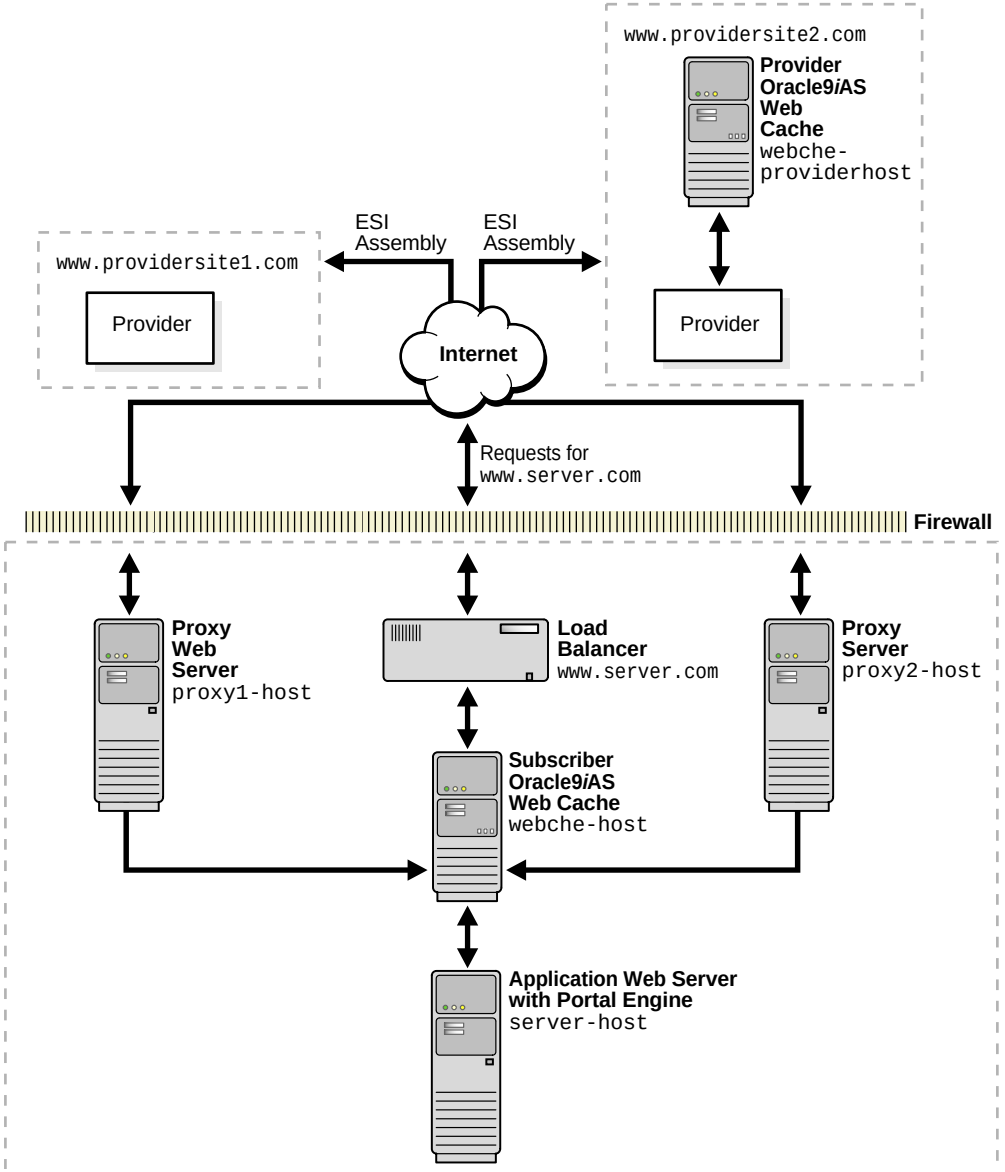


Figure 4–12 Configuring Support for Multiple External ESI Provider Sites

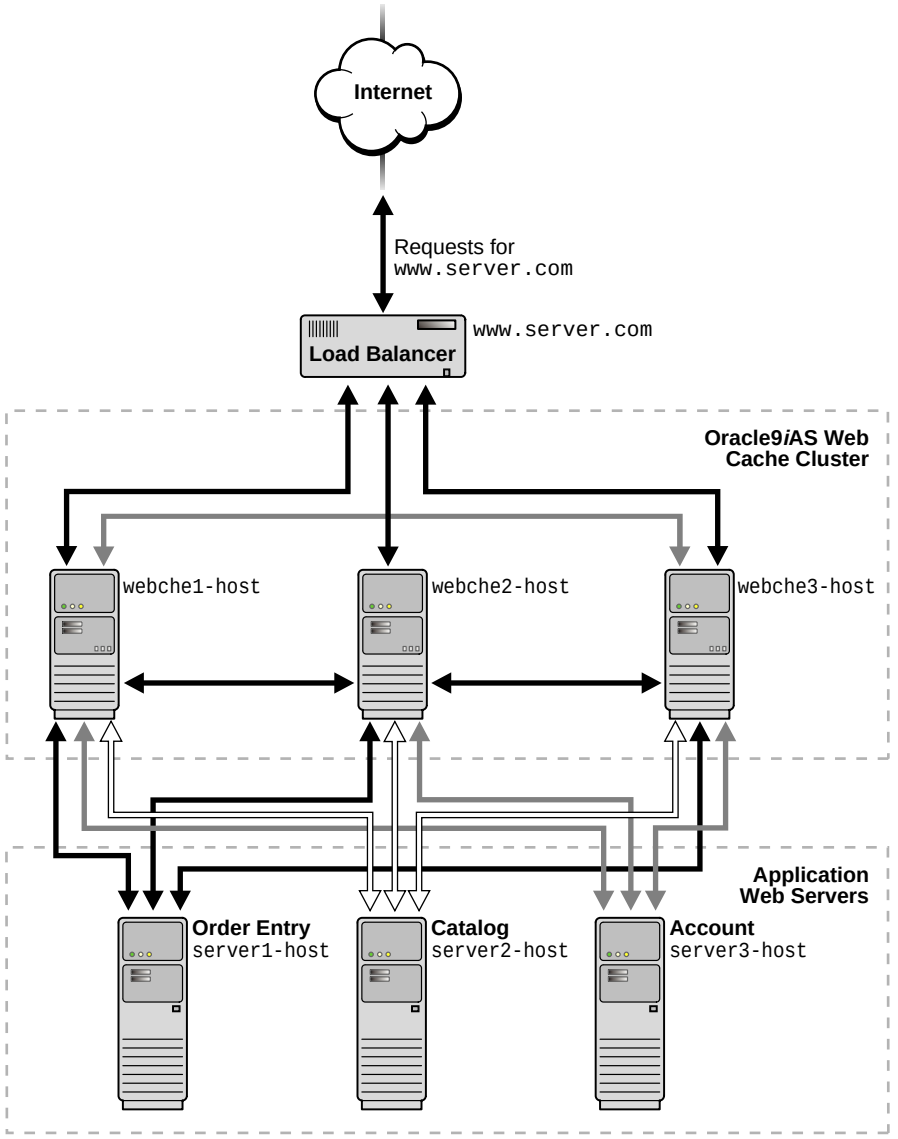


Using Oracle9*i*AS Web Cache Clusters to Increase Availability

To increase the availability and capacity of a Web site, you can configure a cache cluster consisting of two or more Oracle9*i*AS Web Cache servers. Cache clusters support failure detection and failover of Oracle9*i*AS Web Cache servers. If a Oracle9*i*AS Web Cache server fails, other members of the cache cluster detect the failure and take over ownership of the cached content of the failed cluster member. Oracle9*i*AS Web Cache maintains a virtual single cache of content despite a cache failure.

As Figure 4–13 shows, a Load Balancer distributes the requests among the cluster members. The cache cluster members process the incoming requests.

Figure 4–13 Configuring an Oracle9iAS Web Cache Cluster



To configure this deployment:

1. Register the IP address of the Load Balancer with `www.server.com`.
2. Configure the Load Balancer with the host names and capacities of the Oracle9*i*AS Web Cache servers.
3. Configure the Oracle9*i*AS Web Cache cluster, including the three Oracle9*i*AS Web Cache servers in the cluster. In the configuration, specify the host names of the application Web servers and the capacity of each.

Note that many of the deployment scenarios described in this chapter can use a cache cluster in place of one Oracle9*i*AS Web Cache server.

Working with Firewalls

You can deploy Oracle9*i*AS Web Cache inside or outside a firewall.

Figure 4–14 on page 4-26 shows Oracle9*i*AS Web Cache positioned inside a firewall. Deploying Oracle9*i*AS 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 4–14 Configuring Oracle9iAS Web Cache Inside a Firewall

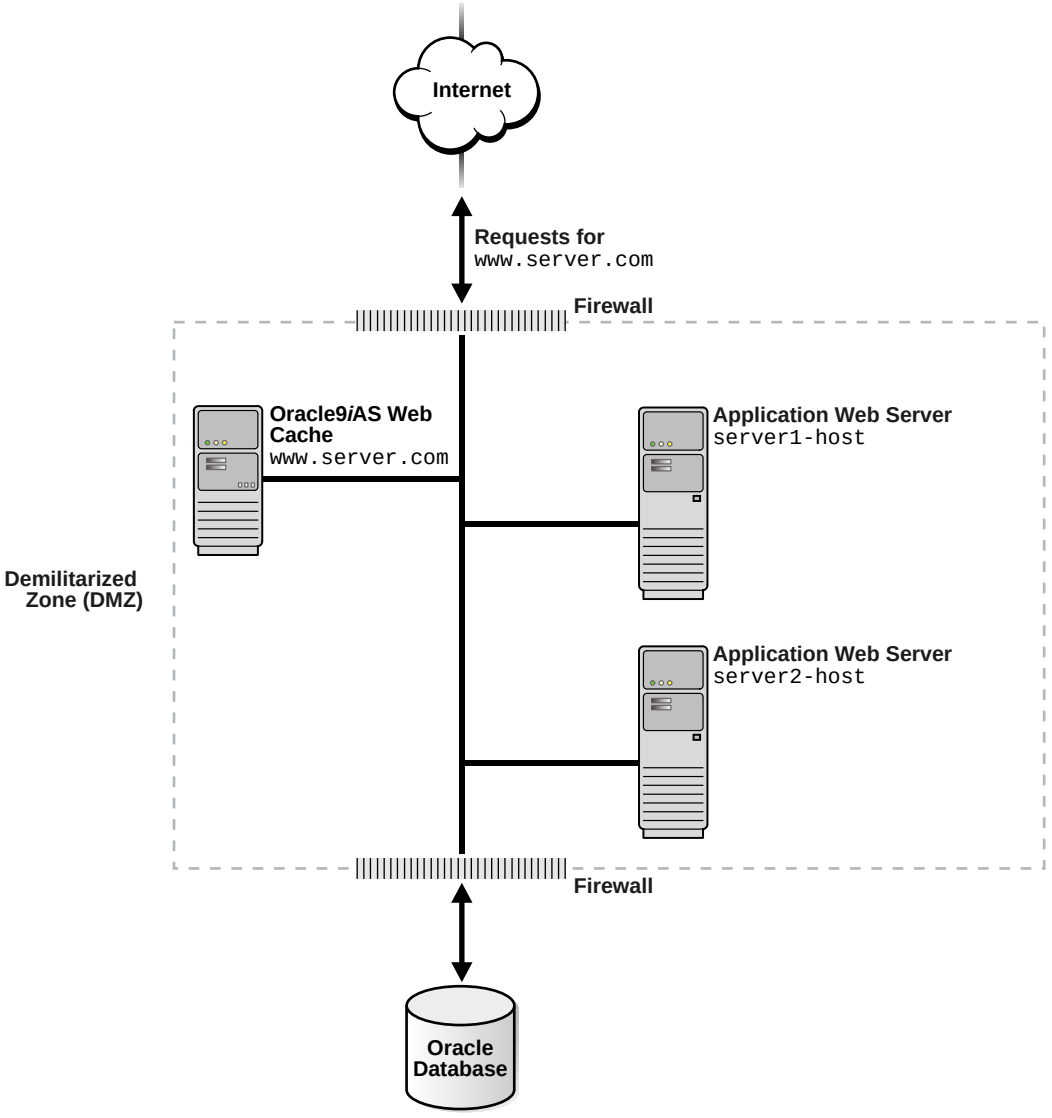
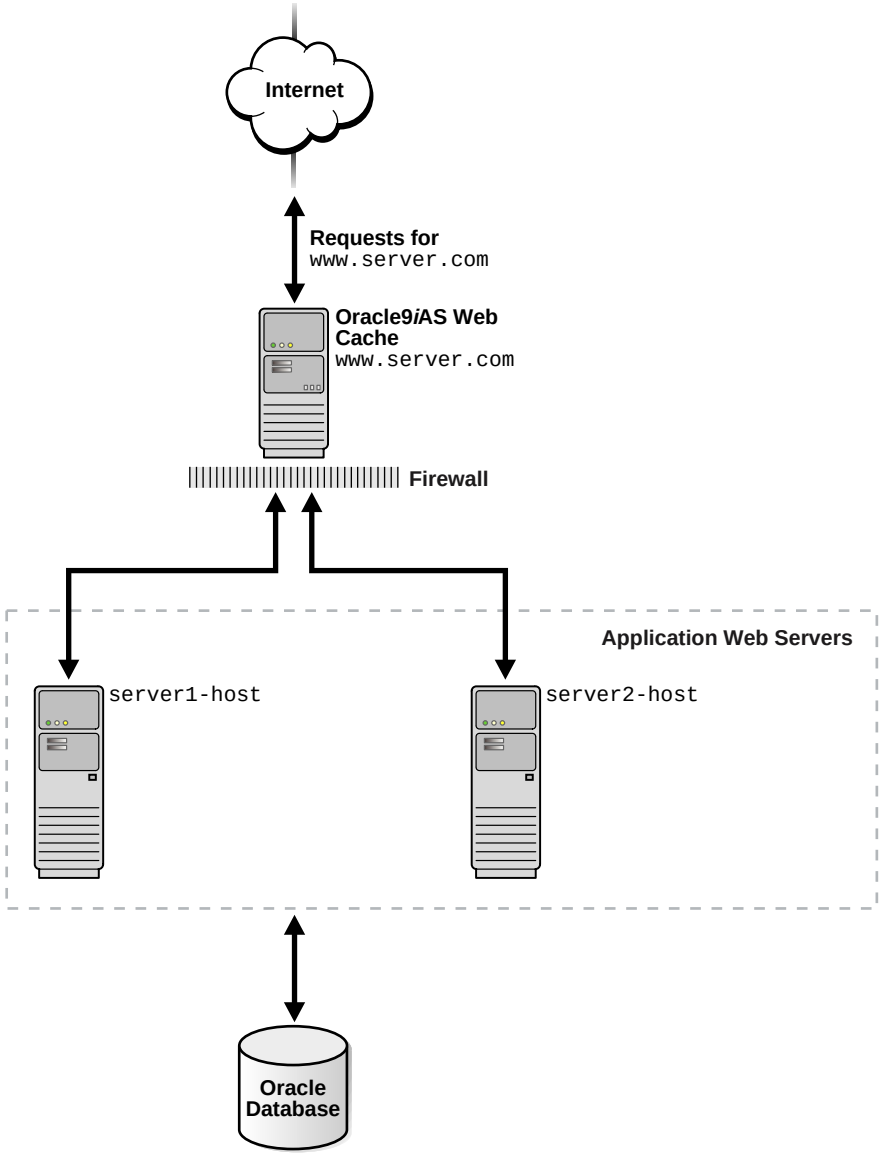


Figure 4–15 on page 4-28 shows Oracle9*i*AS Web Cache positioned outside a firewall. With this deployment, the throughput burden is placed on Oracle9*i*AS Web Cache rather than the firewall. The firewall receives only requests that must go to the application Web servers. This deployment requires securing Oracle9*i*AS 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 Oracle9*i*AS Web Cache outside the DMZ.

Figure 4–15 Configuring Oracle9iAS Web Cache Outside a Firewall



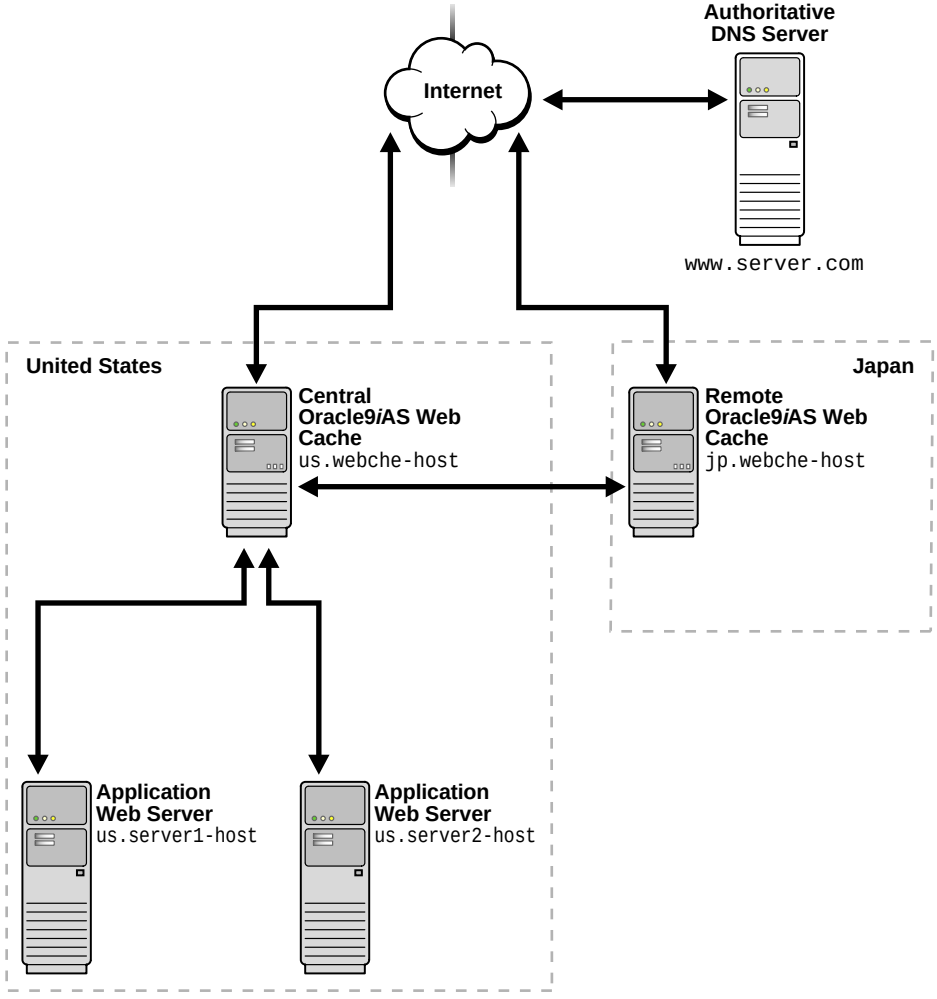
Deploying a Distributed Cache Hierarchy

See Also: "Using Oracle9iAS Web Cache to Support Multiple Sites" on page 4-16 for an example of an ESI cache hierarchy

Many Web sites have several data centers. For networks with a distributed topology, you can deploy Oracle9*i*AS Web Cache at each of the data centers in a **distributed cache hierarchy**. Figure 4–16 on page 4-30 shows a distributed topology in which Oracle9*i*AS Web Cache is distributed in offices in the United States and Japan. The application Web server is located in the United States office, centralizing the data source to one geographic location. The **central cache** in United States caches content for an application Web server, and the **remote cache** in Japan caches content from the central cache.

Browsers make requests 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 to pick the closest Oracle9*i*AS Web Cache server to satisfy the request. It then returns the IP address of the appropriate Oracle9*i*AS Web Cache server to the browser.

Figure 4–16 Deploying an Oracle9iAS Web Cache Hierarchy



To configure this deployment:

1. Configure the local DNS servers with the location of authoritative DNS server `www.server.com`.
2. Configure the authoritative DNS server with the host names and IP addresses of the Oracle9*i*AS Web Cache servers throughout the distributed network.

3. Configure Oracle9*i*AS Web Cache server `us.webche-host` with the following:
   - `us.server1-host` and `us.server2-host` as the application Web servers
   - `www.server.com` as a virtual host site mapped to `us.server1-host` and `us.server2-host`
4. Configure Oracle9*i*AS Web Cache server `jp.webche-host` with the following.
   - `us.webche-host` as the application Web server
   - `www.server.com` as a virtual host site mapped to `us.webche-host`

See Also: "Configuring a Distributed Cache Hierarchy" on page 6-48 for more details about this configuration

5

Configuration and Administration Tools Overview

This chapter introduces the various administration tools of Oracle9*i*AS 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:

Oracle9*i*AS Web Cache Manager for Configuration and Management

Oracle9iAS Web Cache Manager is a graphical user interface tool that combines configuration and monitoring options to provide an integrated environment for configuring and managing Oracle9*i*AS Web Cache and the Web sites for which it caches content. With Oracle9*i*AS Web Cache Manager, you can easily:

- Configure Oracle9*i*AS Web Cache to cache for application Web servers
- Start and stop Oracle9*i*AS Web Cache
- Establish caching rules
- Monitor Oracle9*i*AS Web Cache and Web site performance
- Establish listening ports and security passwords

This section introduces you to the features of Oracle9*i*AS Web Cache Manager. However, the primary documentation for using Oracle9*i*AS Web Cache Manager is the accompanying online help. This section contains these topics:

Starting Oracle9*i*AS Web Cache Manager

To start Oracle9*i*AS Web Cache Manager:

1. Start the **admin server process** with the `webcachectl start` or `webcachectl startadm` commands.

   See Also: "webcachectl Utility for Process Administration" on page 5-9

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 Oracle9*i*AS Web Cache Manager, information about examples, user documentation, and the Oracle Technology Network.

   See Also:

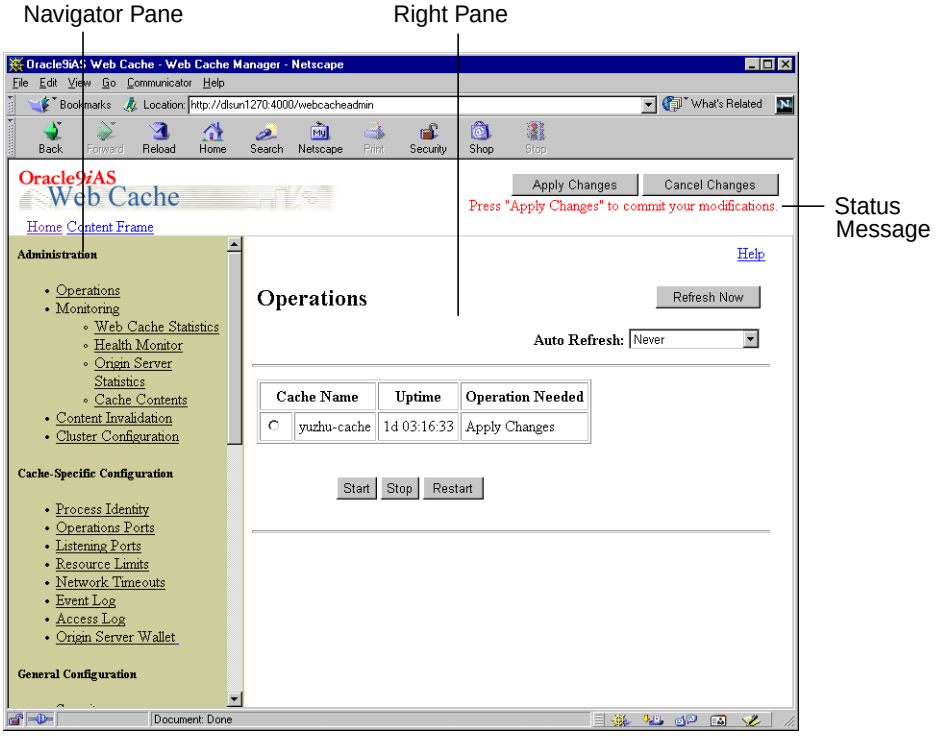
   - "Task 5: Configure Oracle9iAS Web Cache with Operations Ports" on page 6-9 for information on modifying port 4000
   - "Task 2: Modify Security Settings" on page 6-3 for information on modifying the administrator's password

Navigating Oracle9*i*AS Web Cache

The Oracle9*i*AS Web Cache Manager interface includes:

- Top menu bar containing **Apply Changes** and **Cancel Changes** buttons and Oracle9*i*AS Web Cache status message
- Navigator pane with configuration and monitoring menu items
- Right pane with property sheet for selected menu item

Figure 5–1 Oracle9iAS Web Cache Manager Interface



Apply Changes and Cancel Changes Buttons

The **Apply Changes** button applies submitted configuration changes to Oracle9*i*AS Web Cache. The **Cancel Changes** button cancels submitted configuration changes to Oracle9*i*AS Web Cache.

> **Note:** Applied configuration changes require stopping and then restarting Oracle9*i*AS Web Cache. See "Starting and Stopping Oracle9iAS Web Cache" on page 8-2 for further information.

Status Messages

Status messages appear beneath the **Apply Changes** and **Cancel Changes** buttons. Table 5–1 describes the possible status messages.

Table 5–1 Oracle9iAS Web Cache Manager Status Messages

Message	Description
`Web Cache running with current configuration.`	This message appears if Oracle9*i*AS 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 Oracle9iAS Web Cache to make configuration changes take effect`	This message appears if Oracle9*i*AS Web Cache is running with an older version of the configuration. This can happen if configuration changes have been applied but Oracle9*i*AS Web Cache has not been restarted.

In addition, information about any needed operations is displayed in the Operations page.

Navigator Pane

The navigator pane provides a graphical tree view of configuration, administration, and performance monitoring capabilities for Oracle9*i*AS Web Cache and its supported Web sites.

The navigator pane contains the following major folders:

- Administration

  Contains pages that enable you to:

  - Start, stop, and restart Oracle9*i*AS Web Cache
  - Monitor the performance of Oracle9*i*AS Web Cache
  - Monitor the performance of origin servers
  - Invalidate documents in the cache
  - View the contents of a cache
  - Configure a cache cluster

- Cache-Specific Configuration

  Contains pages that enable you to:

  - Specify the process identity for Oracle9*i*AS Web Cache processes
  - Configure Oracle9*i*AS Web Cache listening ports
  - Configure listening ports for administration, statistics, and invalidation requests
  - Specify the storage size of the cache
  - Configure event logging settings
  - Configure access logging settings
  - Specify the location of the wallet used for communication to an **origin server**

- General Configuration

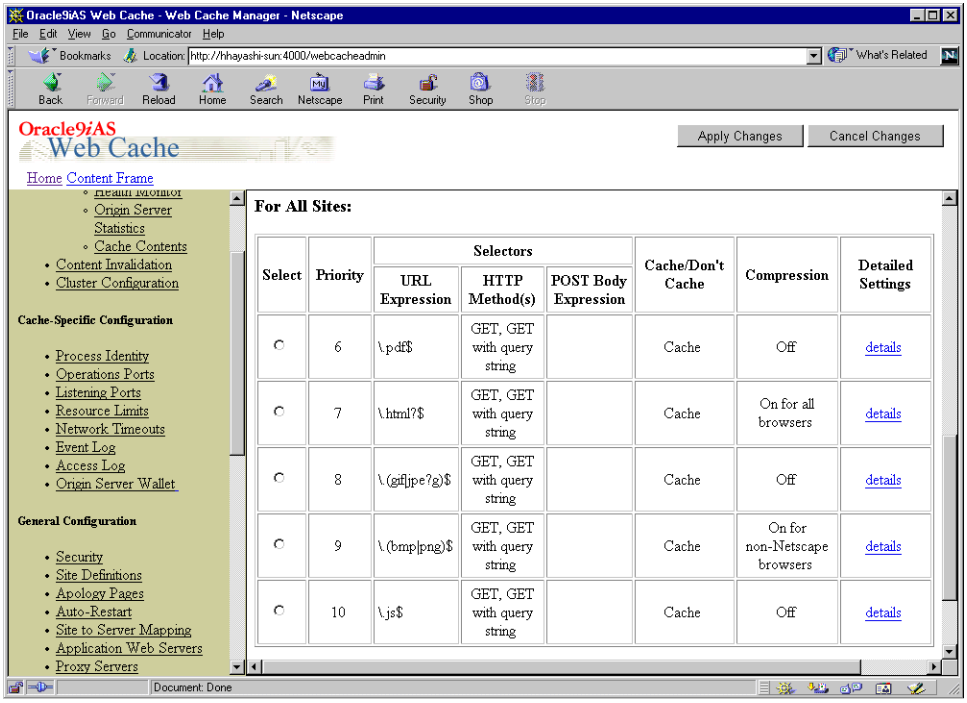
  Contains pages that enable you to:

  - Specify settings for the **auto-restart process**
  - Configure security settings such as usernames and passwords for administrators and invalidation users
  - Specify origin servers
  - Configure site definitions
  - Map site definitions to origin servers
  - Configure caching rules
  - Configure session-related settings

Right Pane

The right pane contains property sheets that enable you to configure and administer Oracle9*i*AS Web Cache. Figure 5–2 shows the Cacheability Rules property sheet used for viewing caching rules.

Figure 5–2 Cacheability Rules Property Sheet



The Operations Page

The Operations page of Oracle9*i*AS Web Cache Manager (**Administration > Operations**) provides information about the status of a cache and what operations are needed. From this page, you can start, stop, or restart a cache.

If the cache is part of a cache cluster, all caches in the cluster are listed on the Operations page. In addition to starting, stopping, and restarting a cache, you can propagate the configuration to other cluster members from this page. You can perform the operations on a selected cache or on all caches in the cluster. To minimize disruption in your Web site, you can specify an interval to stagger the times that the operations begin on the caches.

Oracle Enterprise Manager for Metrics

Oracle Enterprise Manager is the primary tool for administering Oracle9*i* Application Server. It is a complete management solution for administering, configuring, and monitoring the application server and its components. Oracle Enterprise Manager provides a Web-based tool that enables you to view the overall status of Oracle9*i*AS Web Cache and view Oracle9*i*AS Web Cache performance metrics.

Based on the metrics, you can make changes to the configuration with Oracle9*i*AS Web Cache Manager. Later, you can use Enterprise Manager to monitor how configuration changes affect the performance of the Oracle9*i*AS Web Cache.

> **See Also:** *Oracle9i Application Server Administrator's Guide* for further information about using Oracle Enterprise Manager to monitor Oracle9*i*AS Web Cache

webcachectl Utility for Process Administration

The `webcachectl` utility enables you to administer the Oracle9*i*AS Web Cache processes, including the `admin` server process, **cache server process**, and `auto-restart` process:

- The `admin` server process manages the Oracle9*i*AS Web Cache Manager interface

  On Windows, the `admin` server process is represented by the `Oracle`*`HOME_NAME`*`WebCacheAdmin` service

- The `cache` server process manages the cache

  On Windows, the `cache` server process is represented by the `Oracle`*`HOME_NAME`*`WebCache` service

- If enabled, the `auto-restart` process checks that the `cache` server process is running and automatically restart the `cache` server process if it is not running

  Because the `auto-restart` process is dependent upon the `cache` server process, you administer it by starting, stopping, or restarting the `cache` server process.

  On Windows, the `auto-restart` process is represented by the `Oracle`*`HOME_NAME`*`WebCacheMon` service

  > **See Also:** "Task 6: Configure Auto-Restart Process Settings" on page 6-12 for instructions on enabling auto-restart

On UNIX, `webcachectl` must run as the `root` user in the following cases:

- Privileged port numbers less than 1024 are being used for Oracle9*i*AS Web Cache listening ports
- There are more than 1,024 file descriptors being used for connections to Oracle9*i*AS Web Cache.
- The current `webcachectl` user does not match the configured user in the Process Identity page (**Cache-Specific Configuration** > **Process Identity**) of Oracle9*i*AS Web Cache Manager

In order for `webcachectl` to run as the `root` user, ensure that `root.sh` was run during installation. If `root.sh` was not run during installation, run it now from the `$ORACLE_HOME` directory.

On Windows, `webcachectl` runs as the owner of `Oracle`*`HOME_NAME`*`WebCache` service.

The general syntax for this utility follows:

`webcachectl` *`command parameter`*

`webcachectl` is located in the `$ORACLE_HOME/webcache/bin` directory on UNIX and the *`ORACLE_HOME`*`\bin` directory on Windows.

The possible commands for the `webcachectl` utility are listed in Table 5–2. The `start`, `stop`, and `restart` commands enable you to administer all three processes. You can save system resources by administering only the processes you require.

- The `*adm` commands enable you to administer the `admin` server process

  The `admin` server process is the only process required during configuration with the Oracle9*i*AS Web Cache Manager.

- The `*cache` commands enable you to administer the `cache` server process, and if enabled, the `auto-restart` process

  Once Oracle9*i*AS Web Cache Manager configuration is complete, the `admin` server process is no longer needed. The `cache` server process is the only process required to run the cache.

Table 5–2 webcachectl Commands

Command	Description

`reset`	Restores the configuration to the last version saved with **Apply Changes** button in the Oracle9*i*AS Web Cache Manager. The following message displays:
	`Previous configuration restored.`
	`You must restart Oracle Web Cache for it to run with that configuration.`
	`Web Cache admin server is already down.`
	`Web Cache auto-restart monitor is already down.`
	`Web Cache cache server is already down.`
	This command also stops any running processes.

Table 5–2 (Cont.) webcachectl Commands

Command	Description

`restart`	Stops and then restarts the `admin` server, `cache` server, and, if enabled, `auto-restart` processes. The following message displays:
	`Web Cache admin server stopping.`
	`Web Cache auto-restart monitor stopping.`
	`Web Cache cache server stopping.`
	
	`Oracle9iAS Web Cache, Version 9.0.2.0.0`
	`Copyright: Oracle Corporation, 1999-2002`
	`Admin Server now running as process 17722`
	
	`Admin Server is attempting to start the Cache Server`
	
	`Oracle9iAS Web Cache, Version 9.0.2.0.0`
	`Copyright: Oracle Corporation, 1999-2002`
	`Cache Server now running as process 17724`

`restartadm`	Stops and then restarts the `admin` server process. The following message displays:
	`Web Cache admin server is already down.`
	
	`Oracle9iAS Web Cache, Version 9.0.2.0.0`
	`Copyright: Oracle Corporation, 1999-2002`
	`Admin Server now running as process 17729`
	
	`Admin Server running in admin-only mode. Cache Server NOT started`

`restartcache`	Stops and then restarts the `cache` server process and, if enabled, the `auto-restart` process. The following message displays:
	`Web Cache auto-restart monitor is already down.`
	`Web Cache cache server is already down.`
	
	`Oracle9iAS Web Cache, Version 9.0.2.0.0`
	`Copyright: Oracle Corporation, 1999-2002`
	`Cache Server now running as process 17731`

Table 5–2 (Cont.) webcachectl Commands

Command	Description

`start`	Starts the `admin` server, `cache` server process, and, if enabled, the `auto-restart` process. The following message displays:
	`Oracle9iAS Web Cache, Version 9.0.2.0.0`
	`Copyright: Oracle Corporation, 1999-2002`
	`Admin Server now running as process 17736`
	
	`Admin Server is attempting to start the Cache Server`
	
	`Oracle9iAS Web Cache, Version 9.0.2.0.0`
	`Copyright: Oracle Corporation, 1999-2002`
	`Cache Server now running as process 17738`

`startadm`	Starts the `admin` server process. The following message displays:
	`Oracle9iAS Web Cache, Version 9.0.2.0.0`
	`Copyright: Oracle Corporation, 1999-2002`
	`Admin Server now running as process 17745`
	
	`Admin Server running in admin-only mode. Cache Server NOT started`

`startcache`	Starts the `cache` server process and, if enabled, the `auto-restart` process. The following message displays:
	`Oracle9iAS Web Cache, Version 9.0.2.0.0`
	`Copyright: Oracle Corporation, 1999-2002`
	`Cache Server now running as process 17752`

Table 5–2 (Cont.) webcachectl Commands

Command	Description

`status`	Provides running or not running status of the `admin` server process, `cache` server process, and, if enabled, the `auto-restart` process.
	The following message displays when all three processes are not running:
	`Web Cache admin server is not running.`
	`Web Cache auto-restart monitor is not running.`
	`Web Cache cache server is not running.`
	The following message displays when all three processes are running:
	`Oracle Web Cache admin server is running as process 10048.`
	`Oracle Web Cache auto-restart monitor is running as process 10052.`
	`Oracle Web Cache cache server is running as process 10050.`

`stop`	Stops the `admin` server process, `cache` server process, and, if enabled, the `auto-restart` process. The following message displays:
	`Web Cache admin server stopping.`
	`Web Cache auto-restart monitor stopping.`
	`Web Cache cache server stopping.`

`stopadm`	Stops the `admin` server process. The following message displays: `Web Cache admin server stopping.`

`stopcache`	Stops the `cache` server process and, if enabled, the `auto-restart` process. The following message displays:
	`Web Cache auto-restart monitor stopping.`
	`Web Cache cache server stopping.`

The possible parameters for the `webcachectl` utility commands are listed in Table 5–3. These parameters are intended for Oracle Support Services.

Table 5–3 webcachectl Parameters

Command	Description
`coreok`	Enables Oracle9*i*AS Web Cache to produce a core dump

Table 5–3 (Cont.) webcachectl Parameters

Command	Description
`rootmode`	Forces `webcachectl` to run as the `root` user

See Also:

- "Task 6: Configure Auto-Restart Process Settings" on page 6-12 to enable the `auto-restart` process
- "Task 2: Modify Security Settings" on page 6-3 to specify the user and group ID of the Oracle9*i*AS Web Cache executables
- "Starting and Stopping Oracle9iAS Web Cache" on page 8-2 to start or stop Oracle9*i*AS Web Cache

Oracle9*i*AS Web Cache Configuration Files

Oracle9*i*AS Web Cache uses two configuration files: `webcache.xml` and `internal.xml`. Default configuration information created during installation and modify by the Oracle9*i*AS Web Cache Manager is stored in the `webcache.xml` file. Internal configuration settings for Oracle9*i*AS Web Cache are stored in `internal.xml` file. These files are located in the `$ORACLE_HOME/webcache` directory on UNIX and *`ORACLE_HOME`*`\webcache` directory on Windows. Do not edit these configuration files manually, except in the cases described in this guide, or when directed to do so by Oracle Support Services. Improper editing of these configuration files may cause problems in Oracle9*i*AS Web Cache.

Configuration and Administration Tasks at a Glance

Oracle9*i*AS Web Cache configuration and administration tasks are described throughout this guide and in the Oracle9*i*AS Web Cache Manager online help system. Table 5–4 lists the common tasks, and points you to the topic in this guide that describes the task.

Table 5–4 Common Administrative Tasks for Oracle9iAS Web Cache

Task	See Also
Configuring Oracle9*i*AS Web Cache	
Change the `administrator` or `invalidator` password.	"Task 2: Modify Security Settings" on page 6-3
Enable the `auto-restart` process.	"Task 6: Configure Auto-Restart Process Settings" on page 6-12
Modify the network time-outs for Oracle9*i*AS Web Cache.	"Task 7: Configure Network Time-outs" on page 6-13
Specify the settings for origin servers.	"Task 9: Specify Settings for Origin Servers" on page 6-23
Configure Web site definitions and map the site to origin servers.	"Task 10: Configure Web Site Settings" on page 6-26
Configure Oracle9*i*AS Web Cache with listening ports.	"Task 3: Configure Oracle9iAS Web Cache with Listening Ports for Incoming Browser Requests" on page 6-6
Modify listening ports for administration, invalidation, and statistics monitoring requests.	"Task 5: Configure Oracle9iAS Web Cache with Operations Ports" on page 6-9
Set the maximum cache size limit.	"Task 11: Specify Caching Rules" on page 6-37
Configure caching rules.	"Configuring Caching Rules" on page 7-8
Load balance requests over multiple origin servers.	"Configuring Load Balancing and Failover" on page 6-45
Bind a session to an origin server.	"Binding a Session to an Origin Server" on page 6-45
Configure a cache hierarchy.	"Configuring a Hierarchy of Caches" on page 6-48
Configure a cache cluster.	"Configuring a Cache Cluster" on page 6-55
Configure event log settings.	"Configuring Event Logs" on page 8-56
Configure access log settings.	"Configuring Access Logs" on page 8-64

Table 5–4 (Cont.) Common Administrative Tasks for Oracle9iAS Web Cache

Task	See Also
Configuring Support for HTTPS Requests	
Create a wallet.	"Task 1: Create Wallets" on page 6-39
Configure Oracle9*i*AS Web Cache with an HTTPS listening port.	"Task 3: Configure Oracle9iAS Web Cache with Listening Ports for Incoming Browser Requests" on page 6-6
Configure HTTPS listening ports for administration, invalidation, and statistics monitoring requests.	"Task 5: Configure Oracle9iAS Web Cache with Operations Ports" on page 6-9
Configure the origin server with an HTTPS listening port.	"Task 9: Specify Settings for Origin Servers" on page 6-23
Administering Oracle9*i*AS Web Cache	
Start and stop Oracle9*i*AS Web Cache	"Starting and Stopping Oracle9iAS Web Cache" on page 8-2
Invalidate documents in the cache.	"Invalidating Documents in the Cache" on page 8-6
Propagate configuration changes to cache cluster members.	"Propagating Configuration Changes to Cache Cluster Members" on page 8-5
List the URLs of the documents in the cache.	"Listing the Contents of the Cache" on page 8-49
Monitoring Performance	
Monitor overall Oracle9*i*AS Web Cache health.	"Monitoring Oracle9iAS Web Cache Health" on page 9-2
Monitor Oracle9*i*AS Web Cache performance.	"Gathering Oracle9iAS Web Cache Performance Statistics" on page 9-4
Monitor origin server performance.	"Gathering Origin Server Performance Statistics" on page 9-8

Note: All tasks listed under the **Configuring Oracle9*i*AS Web Cache** and **Configuring Support for HTTPS Requests** rows require stopping and then restarting Oracle9*i*AS Web Cache. See "Starting and Stopping Oracle9iAS Web Cache" on page 8-2 for further information.

Part II

Configuration and Administration of Oracle9*i*AS Web Cache

Part II describes how to set up and configure Oracle9*i*AS Web Cache.

This part contains these chapters:

6

Initial Setup and Configuration

This chapter describes the steps to initially configure Oracle9*i*AS Web Cache to begin caching content. It also provides instructions for configuring multiple **origin servers** and a **cache cluster**.

This chapter contains these topics:

Setting Up Oracle9*i*AS Web Cache

To set up Oracle9*i*AS Web Cache, perform the following tasks:

Task 1: Start Oracle9*i*AS Web Cache and the Oracle9*i*AS Web Cache Manager

To start Oracle9*i*AS Web Cache Manager to begin initial configuration:

1. If not currently logged on to the Oracle9*i*AS Web Cache computer, log in with the user ID of the user that performed the installation.
2. Start Oracle9*i*AS Web Cache. From the command line, enter:

   ```
   webcachectl start
   ```

3. Start Oracle9*i*AS Web Cache Manager.

See Also: "Starting Oracle9iAS Web Cache Manager" on page 5-3

Task 2: Modify Security Settings

When Oracle9*i*AS Web Cache is installed, it is set up with default passwords for administration and invalidation requests. In addition, the computer on which you installed Oracle9*i*AS Web Cache is the default trusted host.

To change the security settings:

1. Change the password for the administrator.

   Configuration and operational tasks can be performed with the Oracle9*i*AS 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 **General Configuration** > **Security**.

      The Security page appears.

   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 20 characters long in the **New Password** and **Confirm New Password** fields.

   d. Choose **Submit**.

2. 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 20 characters long in the **New Password** and **Confirm New Password** fields.

   c. Choose **Submit**.

3. Optionally, change the trusted subnet or trusted host from which administration, invalidation, and statistics monitoring requests can take place.

By default, the computer on which you installed Oracle9*i*AS 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 requests from all computers in all the subnets in the network.

This machine only

Select to allow requests from only this computer.

Enter list of IPs

Select to allow 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.

Note: Sometimes requests come through a proxy server. If the proxy server is not covered by the trusted subnet settings, then requests will fail.

---

c. Choose **Submit**.

4. Optionally, change the user ID and group ID for the Oracle9*i*AS Web Cache executables on UNIX.

   By default, the user that performed the installation is the owner of Oracle9*i*AS Web Cache executables. This user can execute `webcachectl` commands. Users that belong to the same group ID of the user that performed installation can also execute `webcachectl` commands.

   To change the user ID and group ID for the Oracle9*i*AS Web Cache executables on UNIX:

   a. In the navigator pane, select **Cache-Specific Configuration** > **Process Identity**.

      The Process Identity page appears.

   b. In the Process Identity page, select a cache for which to modify settings, and then 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**.

   e. Manually change the ownership of the following files and directories to the new user ID and group ID with the `chown` command:

      - `$ORACLE_HOME/webcache`
      - `$ORACLE_HOME/webcache/internal.xml`
      - `$ORACLE_HOME/webcache/internal_admin.xml`
      - `$ORACLE_HOME/webcache/webcache.xml`
      - `$ORACLE_HOME/webcache/logs/event_log`
      - `$ORACLE_HOME/webcache/logs/access_log`

5. In the Oracle9*i*AS Web Cache Manager main window, choose **Apply Changes**.

Note: If you changed any of the security settings, you must restart the **admin server process** from the `webcachectl` utility rather than with the **Restart** option in the Operations page (**Administration** > **Operations**). See "Task 12: Apply Changes and Restart Oracle9iAS Web Cache" on page 6-37.

See Also: "Administrator Password in the Change Administration Password Dialog Box" on page 10-15

Task 3: Configure Oracle9*i*AS Web Cache with Listening Ports for Incoming Browser Requests

By default, Oracle9*i*AS Web Cache listens with the HTTP protocol on port 7777 and HTTPS on port 4443. If these ports are in use, then the installation procedure attempts to assign other port numbers from a range of possible port numbers.

Note: The IP addresses for the default HTTP and HTTPS ports are set to ANY. Upon startup, Oracle9*i*AS Web Cache attempts to bind the ports to all IP addresses. If multiple instances of Oracle9*i*AS Web Cache are running on a multihomed host with multiple IP addresses, then change ANY to a specific IP address to avoid port conflicts in the Listening Ports page (**Cache-Specific Configuration** > **Listening Ports**.

It may be necessary to add an additional listening port if you want to assign Oracle9*i*AS Web Cache a port that an origin server was previously listening on.

To specify a listening port from which Oracle9*i*AS Web Cache can receive browser requests:

1. In the navigator pane, select **Cache-Specific Configuration** > **Listening Ports**.

   The Listening Ports page appears.

2. In the Listening Ports page, choose **Add**.

   The Edit Listening Ports page dialog box appears.

3. From the list, select the cache for which to modify settings.

4. In the **IP Address** field, enter the IP address of the computer running Oracle9*i*AS Web Cache.

5. In the **Port Number** field, enter the listening port from which Oracle9*i*AS Web Cache will receive Web browser requests for the Web site.

   Ensure that this port number is not already in use.

6. 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.

See Also: "Secure Sockets Layer (SSL) Support" on page 1-27

7. If you selected HTTPS as the listening protocol, then enter the location of the **wallet** in the **Wallet** field.

   This wallet is used for browser requests for sites hosted by Oracle9*i*AS Web Cache.

   By default, wallets are stored in the following locations:

   - `/etc/ORACLE/WALLETS/`*`user_name`* on UNIX
   - `%USERPROFILE%\ORACLE\WALLETS` on Windows operating systems

   Oracle Corporation recommends entering the location, even if the default is being used.

   As long as each site is configured with a separate wallet, the Oracle9*i*AS Web Cache listening port can share the same wallet as specified in the Origin Server Wallet page (**Cache-Specific Configuration** > **Origin Server Wallet**).

8. Choose **Submit**.

See Also:

- "Task 9: Specify Settings for Origin Servers" on page 6-23 for instructions on specifying the origin server wallet
- "Configuring Oracle9iAS Web Cache for HTTPS Requests" on page 6-38 for complete instructions on HTTPS configuration

Task 4: Provide Directives to Oracle HTTP Server

At installation time, Oracle HTTP Server sets the `httpd.conf` file with the following directives that impact Oracle9*i*AS Web Cache:

- `Port=`*`web_cache_port`* specifies the Oracle9*i*AS Web Cache listening ports, enabling dynamically created URLs to be redirected to Oracle9*i*AS Web Cache
- `Listen=`*`Oracle_HTTP_Server_port`* specifies the HTTP and HTTPS ports obtained by Oracle HTTP Server
- `ServerName` specifies the host name of Oracle HTTP Server
- `UseCanonicalName On` instructs Oracle HTTP Server to use the host names and port values set in the `ServerName` and `Port` directives when redirecting a URL

For example:

```
##
## httpd.conf -- Apache HTTP server configuration file
##
...
Port 7777
Listen 7778
...
ServerName http_server.company.com
...
UseCanonicalName On
....
```

If you decide to disable Oracle9*i*AS Web Cache, then the Oracle HTTP Server administrator must modify the value of the `Port` directive to the same value set for the `Listen` directive. For example:

```
##
## httpd.conf -- Apache HTTP server configuration file
##
...
Port 7778
Listen 7778
...
ServerName http_server.company.com
...
UseCanonicalName On
....
```

If Oracle9*i*AS Web Cache is deployed on a separate computer from the Oracle HTTP Server, then the Oracle HTTP Server administrator must modify the `ServerName` directive in `httpd.conf` for each site hosted by Oracle9*i*AS Web Cache. This will enable Oracle HTTP Server to redirect URLs to Oracle9*i*AS Web Cache. The following example shows `httpd.conf` modified to direct requests for `www.1st.company.com` and `www.2nd.company.com` to Oracle9*i*AS Web Cache, which listening on port 7777.

```
Port 7777
Listen 7778
...
ServerName  www.1st.company.com
ServerName  www.2nd.company.com
...
UseCanonicalName On
....
```

The `httpd.conf` file resides in `$ORACLE_HOME/Apache/Apache/conf/httpd.conf` on UNIX or *`ORACLE_HOME`*`\Apache\Apache\conf\httpd.conf` on Windows.

See Also: *Oracle HTTP Server Administration Guide*

Task 5: Configure Oracle9*i*AS Web Cache with Operations Ports

In addition to receiving HTTP and HTTPS browser requests, Oracle9*i*AS 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, Oracle9*i*AS Web Cache uses the HTTP protocol to receive these requests. Default HTTP port numbers are as follows:

- 4000 for administration and configuration requests from Oracle9*i*AS Web Cache Manager
- 4001 for invalidation requests
- 4002 for statistics monitoring requests

Note: The default configuration does not enable HTTPS for administration, invalidation, or statistics monitoring requests. Instead, these ports are configured for HTTP basic authentication. The passwords for the `administrator` user and the `invalidator` user can be decoded when they are sniffed out of the HTTP traffic. To avoid breach of security information for unprotected and insecure networks, modify the protocol to HTTPS to ensure that the passwords for these requests are secure. Perform the procedure that follows.

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

1. In the navigator pane, select **Cache-Specific Configuration** > **Operations Ports**.

   The Operations Ports page appears.

2. Select the cache for which to modify port and protocol settings.
3. In the Operations Ports page, choose **Edit**.

   The Edit Operations Port dialog box appears.

4. In the **ADMINISTRATION**, **INVALIDATION**, or **STATISTICS** row, perform the following:

   a. Enter the new port in the **Port Number** field.

   b. From the **Protocol** list, select either **HTTP** or **HTTPS** to accept requests.

   c. If you selected HTTPS, then enter the location of the wallet in the **Wallet** field.

      This wallet is used for administration, invalidation, and statistics monitoring HTTPS requests to Oracle9*i*AS Web Cache.

      By default, wallets are stored in the following locations:

      – `/etc/ORACLE/WALLETS/`*`user_name`* on UNIX

      – `%USERPROFILE%\ORACLE\WALLETS` on Windows operating systems

      Oracle Corporation recommends entering the location, even if the default is being used.

As long as each site is configured with a separate wallet, these ports can share the same wallet as specified in the Listening Ports page (**Cache-Specific Configuration** > **Listening Ports**) and the Origin Server Wallet page (**Cache-Specific Configuration** > **Origin Server Wallets**).

5. Choose **Submit**.

See Also:

- "Task 9: Specify Settings for Origin Servers" on page 6-23 for instructions on specifying the origin server wallet location
- "Task 3: Configure Oracle9iAS Web Cache with Listening Ports for Incoming Browser Requests" on page 6-6 for instructions on specifying theOracle9*i*AS Web Cache wallet location
- "Configuring Oracle9iAS Web Cache for HTTPS Requests" on page 6-38 for complete instructions on HTTPS configuration

Notes:

- 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 (**General Configuration** > **Security**). See "Task 2: Modify Security Settings" on page 6-3 for further information.
- If you changed any of the operations ports, you must restart the `admin` server process from `webcachectl` utility rather than with the **Restart** option in the Operations page (**Administration** > **Operations**). See "Task 12: Apply Changes and Restart Oracle9iAS Web Cache" on page 6-37.

Task 6: Configure Auto-Restart Process Settings

The **auto-restart process** checks that the **cache server process** is running and automatically restarts the `cache` server process if it is not running.

Note: On Windows, the `cache` server process is represented by the `Oracle`*`HOME_NAME`*`WebCache` service, and the `auto-restart` process is represented by the `Oracle`*`HOME_NAME`*`WebCacheMon` services.

If you enable auto-restart, then the `auto-restart` process polls the Oracle9*i*AS Web Cache server at specified intervals. It does this by sending requests to a specified URL. If it cannot connect to the `cache` server or if the `cache` server does not respond within a specified time, then the `auto-restart` process restarts the `cache` server process.

By default, auto-restart is not enabled.

To specify the settings for auto-restart:

1. Start Oracle9*i*AS Web Cache Manager.

   See Also: "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. In the navigator pane, select **General Configuration** > **Auto-Restart.**

   The Auto-Restart page appears.

3. To change the default settings, choose **Edit.**

   The Edit Auto-Restart dialog box is displayed.

4. To enable Auto-Restart, select **Enabled**.

5. In the **Failover Threshold** field, enter the number of consecutive failed requests before the `auto-restart` process considers the `cache` server to have failed. Only network errors, including timeout errors, are counted as failed requests.

   For each failed request, Oracle9*i*AS Web Cache increments the failure counter. When a request is successfully processed by the `cache` server, Oracle9*i*AS Web Cache resets the failure counter.

   When the failover threshold is met, the `auto-restart` process starts the `cache` server.

Note: The threshold applies only to network errors and time-outs. If the `cache` server process is not running when the `auto-restart` process attempts to poll the `cache` server, the `auto-restart` process immediately restarts the `cache` server.

---

6. In the **Ping URL** field, enter the URL that the `auto-restart` process will use to poll the `cache` server.

   Use a URL that you can guarantee is stored in the cache. The default is "`/`".

7. In the **Ping Interval** field, enter the time, in seconds, between attempts by the `auto-restart` process to poll the `cache` server.

   The default value is 15 seconds.

8. In the **Ping Timeout** field, enter the time, in seconds, that the `auto-restart` process will wait for a response from the `cache` server.

   The default value is 30 seconds.

9. Choose **Submit.**

Task 7: Configure Network Time-outs

After Oracle9*i*AS Web Cache sends a response to a browser, the connection is left open for five seconds, which is typically enough time for the browser to process the response from Oracle9*i*AS Web Cache. If the network between the browser and Oracle9*i*AS Web Cache is slow, consider increasing the time-out. Likewise, there is a 3600 second network time-out between Oracle9*i*AS Web Cache and the origin server. If the origin server is unable to generate a response within that time, Oracle9*i*AS Web Cache sends a network apology page to the browser. If applications require a shorter time-out, then adjust the time-out.

To modify the default network time-outs:

1. In the navigator pane, select **Cache-Specific Configuration** > **Network Timeouts**.

   The Network Timeouts page appears.

2. In the Network Timeouts page, select the cache, and then choose **Edit**.

   The Edit Network Timeouts dialog box appears.

3. In the **Keep-Alive** field, enter the time, in seconds, for Oracle9*i*AS Web Cache to keep a connection open to the browser after it has returned a response.

If the time-out is set to 0, then the connection to the browser is not kept open. In addition, Oracle9*i*AS Web Cache sends the following response-header field in the response:

```
Connection: Close
```

4. In the **Origin Server Timeout** field, enter the time, in seconds, for the origin server to generate a response to Oracle9*i*AS Web Cache.
5. Choose **Submit**.

See Also: "Content-Length Request-Header Field" on page 10-13

Task 8: Set Resource Limits

To set resource limits for Oracle9*i*AS Web Cache, configure the following attributes:

- Cache Memory
- Connection Limit

Cache Memory

When the maximum cache memory limit is reached, Oracle9*i*AS Web Cache performs garbage collection. During garbage collection, Oracle9*i*AS Web Cache removes the less popular and less valid documents from the cache in favor of the more popular and more valid documents. In a cache cluster environment, Oracle9*i*AS Web Cache removes on-demand documents before it removes owned 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 Oracle9*i*AS Web Cache should be set to at least 256 MB. By default, the maximum cache size is set to 50 MB, which is sufficient only for initial post-installation testing.

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

1. Determine which documents you want to cache, how many are smaller than 4 KB and how many are larger than 4 KB. Determine the average size of the documents that are larger than 4 KB. Determine the expected peak load—the maximum number of documents to be processed concurrently.

   One way to do this is to look at existing Web server logs for one day to see which 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 Oracle9*i*AS Web Cache.

   Starting with release 9.0.2, the amount of memory that Oracle9*i*AS 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 KB, Oracle9*i*AS Web Cache uses a buffer of 4 KB to store the HTTP body.
   - If a document is 4 KB or larger, Oracle9*i*AS Web Cache uses buffers of 32 KB to store the HTTP body. For example, if a document is 40 KB, Oracle9*i*AS Web Cache uses two 32 KB buffers to store the HTTP body.
   - Regardless of the size of the body, Oracle9*i*AS Web Cache uses 4 KB to store the HTTP response header.

   Starting with release 9.0.2, use the following formula to determine an estimate of the maximum memory needed:

   (*X* * (4KB + 4KB)) + (*Y* * (([*m*/32] * 32KB) + 4KB)) + *basemem*

   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.
   - 4KB 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.
   - 4KB is the size of the buffer for the HTTP response header.
   - *basemem* is the base amount of memory needed by Oracle9*i*AS Web Cache to process requests. This amount includes memory for internal functions such as lookup keys and timestamps. The amount needed depends on the

number of concurrent requests and on whether or not the requests include **Edge Side Includes (ESI)**.

For non-ESI requests, each concurrent request needs approximately 6 KB to 25 KB of memory. For example, to support 1000 concurrent requests, you need between 6 MB and 25 MB of memory.

For ESI requests, each concurrent request needs approximately the following amount of memory:

```
60KB + (number of ESI fragments * [6KB to 25KB])
```

For example, for a document with 10 ESI fragments, use the following calculation:

```
60KB + (10 * [6KB to 25KB]) = 120KB to 330KB
```

That is, you need between 120 KB and 330 KB of memory for one 10-fragment document. To support 1000 concurrent requests, you need approximately between 120 MB to 330 MB of memory.

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. The documents do not use ESI. You expect to process 500 documents concurrently. Use the formula to compute the maximum memory:

```
(5000 * (4KB + 4KB) ) + ( 2000 * (( [54/32] * 32KB ) + 4KB )) + (500 *[6KB to 25KB])
```

Using the formula, you need:

- 40,000 KB for the smaller documents.
- 136,000 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 4 KB for each document.
- 3,000 KB to 12,500 KB for the base amount of memory needed to process 500 concurrent requests.

This results in an estimate of 179,000 KB to 188,500 KB of memory needed.

Note: Even though you specify 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 from the preceding formula.

3. Configure Oracle9*i*AS 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, perform the following steps:

   a. In the navigator pane, select **Cache-Specific Configuration > Resource Limits.**

   b. On the Resource Limits page, select the cache and choose **Edit.**

      The Edit Resource Limits dialog box appears.

   c. In the **Maximum Cache Size** field, enter the result of the formula.

   d. Choose **Submit.**

4. After applying changes and restarting Oracle9*i*AS Web Cache, as described in "Task 12: Apply Changes and Restart Oracle9iAS Web Cache" on page 6-37, use a simulated load or an actual load to monitor the cache to see how much memory it really uses in practice.

   Remember that the cache is empty when Oracle9*i*AS Web Cache starts. For monitoring to be valid, ensure that the cache is fully populated. That is, ensure that the cache has received enough requests so that a representative number of documents are cached.

   The Web Cache Statistics page provides information about the current memory use and the maximum memory use.

To access the Web Cache Statistics page, from the navigator pane, select **Administration > Monitoring** > **Web Cache Statistics**. Note the following metrics in the Cache Overview table:

- **Size of Documents in Cache** 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 **Size of Documents in Cache** is 53 KB, the total of the two sizes. This metric does not show the physical size of the cache.
- **Configured Maximum Cache Size** indicates the maximum cache size as specified in the Resource Limits page.
- **Current Allocated Memory** displays the physical size of the cache. The physical size of the cache is the amount of data memory allocated by Oracle9*i*AS Web Cache for cache storage and operation. This number is always smaller than the process size shown by operating system statistics because the Oracle9*i*AS Web Cache process, like any user process, consumes memory in other ways, such as instruction storage, stack data, thread, and library data.
- **Current Action Limit** is 90% of the **Configured Maximum Cache Size**. This number is usually larger than the **Current Allocated Memory**.

If **Current Allocated Memory** is greater than **Current Action Limit**, Oracle9*i*AS Web Cache begins garbage collection. That is, Oracle9*i*AS Web Cache removes the less popular and less valid documents from the cache in favor of the more popular and more valid documents to obtain space for new HTTP responses without exceeding the maximum cache size.

If the **Current Allocated Memory** is close to or greater than the **Current Action Limit**, increase the maximum cache size to avoid swapping documents in and out of the cache. Use the **Cache-Specific Configuration** > **Resource Limits** page to increase the maximum cache size.

Connection Limit

In addition to the cache size, it is also important to specify a reasonable number for the maximum connection limit for the Oracle9*i*AS Web Cache server. If you set a number that is too high, performance can be affected, resulting in slower response time. If you set a number that is too low, fewer requests will be satisfied. You must strike a balance between response time and the number of requests processed concurrently.

To help determine a reasonable number, consider the following factors:

- The maximum number of clients you intend to serve concurrently at any given time
- The average size of a page and the average number of requests for page
- Network bandwidth. The amount of data that can be transferred at any one time is limited by the network bandwidth.
- The percentage of cache misses. If a large percentage of requests are cache misses, the requests are forwarded to the application Web server. Those requests consume additional network bandwidth and result in longer response times.
- How quickly a page is processed. Use a network monitoring utility, such as `ttcp`, to determine how quickly your system processes a page.
- The cache cluster member capacity, if you have a cache cluster environment. The capacity reflects the number of incoming connections from other cache cluster members. You set the cluster member capacity using the **Administration** > **Cluster Configuration** page of Oracle9*i*AS Web Cache Manager.

Use various tools, such as those available with the operating system and with Oracle9*i*AS Web Cache, to help you determine the maximum number of connections. For example, the `netstat -a` command on UNIX and Windows operating systems enables you to determine the number of established connections; the `ttcp` utility enables you to determine how fast a page is processed. Oracle9*i*AS Web Cache Manager provides statistics on hits and misses.

To set the maximum number of incoming connections, perform the following steps:

1. In the navigator pane of Oracle9*i*AS Web Cache Manager, select **Cache-Specific Configuration > Resource Limits.**
2. In the Resource Limits page, select the cache and choose **Edit**.

   The Edit Resource Limits dialog box appears.
3. In the **Maximum Incoming Connections** field, enter the new value.
4. Choose **Submit**.

Do not set the value to an arbitrary high value, because Oracle9*i*AS Web Cache sets aside some resources for each connection, which could adversely affect performance. For many UNIX systems, 5000 connections is usually a reasonable number.

Connections on UNIX On most UNIX platforms, each client connection requires a separate file descriptor. Oracle9*i*AS Web Cache tries to reserve the maximum number of file descriptors (`Max_File_Desc`) when it starts. As long as `webcachectl` can run as root, you can change this number to a higher one. For example, on Sun Solaris, you can increase the maximum number of file descriptors by setting the `rlim_fd_max` parameter. If the `webcachectl` is not able to run as root, Oracle9*i*AS Web Cache server logs an error message and fails to start.

See Also:

- "webcachectl Utility for Process Administration" on page 5-9 for root user requirements for `webcachectl`
- "Greater Than One Thousand Maximum Connections" on page 10-6

Starting with release 9.0.2, Oracle9*i*AS Web Cache uses the following formula to calculate the maximum number of file descriptors to be used:

```
Max_File_Desc = Curr_Max_Conn + Total_WS_Capacity + Outgoing_Cluster_Conn + 100
```

In the formula:

- `Max_File_Desc` is the maximum number of file descriptors to be used.
- `Curr_Max_Conn` is the current maximum incoming connections limit for Oracle9*i*AS Web Cache. You set the maximum number of incoming connections using the **Cache-Specific Configuration > Resource Limits** page of the Oracle9*i*AS Web Cache Manager.

  In a cache cluster environment, `Curr_Max_Conn` also includes the cluster member capacity, which is the incoming connections from peer caches. You set the capacity using the **Administration > Cluster Configuration** page of the Oracle9*i*AS Web Cache Manager.
- `Total_WS_Capacity` is the sum of the capacity for all configured application Web servers. You set the capacity using the **General Configuration** > **Application Web Servers** page of Oracle9*i*AS Web Cache Manager.

  In a cache cluster environment, the capacity is divided among the cache cluster members, using the following formula:

  ```
  Total_WS_Capacity = Sum_Web_Server_Capacity / n
  ```

  In the formula, `Sum_Web_Server_Capacity` is the sum capacity of all configured application Web servers, and `n` is the number of cache cluster members. For example, assume you have two configured application Web Servers. Web_Server_A has a capacity of 200 and Web_Server_B has a capacity of 250. Also, assume you have a cluster with three caches. The `Total_WS_Capacity` is 150, as the following example calculates:

  ```
  Total_WS_Capacity = (200 + 250) / 3
  ```

- `Outgoing_Cluster_Conn` is the total of outgoing connections to peer caches in a cache cluster. The value is zero if you do not have a cache cluster. To compute this value, use the following formula:

  ```
  Outgoing_Cluster_Conn = Sum_Cluster_Capacity / (n-1)
  ```

  In the formula, `Sum_Cluster_Capacity` is the sum of the capacity of all other caches in a cluster, and `n` is the number of cache cluster members. For example, assume you have cluster with three caches. Cache_A has a capacity of 100, Cache_B has a capacity of 150, and Cache_C has a capacity of 200. The `Outgoing_Cluster_Conn` for Cache_A, is 175, as the following example calculates:

  ```
  Outgoing_Cluster_Conn = (150 + 200) / (3-1)
  ```

To set the capacity of caches in a cluster, select **Administration > Cluster Configuration** from the navigator pane of Oracle9*i*AS Web Cache Manager.

- `100` is the number of connections reserved for internal use by Oracle9*i*AS Web Cache.

See Also:

- Operating system-specific documentation for connection limitations
- *Oracle9i Application Server Performance Guide* for TCP / IP performance tuning tips

Connections on Windows On Windows operating systems, 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 operating systems allow up to 65534 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 65534 - 1024

The total of the maximum number of incoming connections and cluster member capacity should not be set to a number greater than the number of TCP ports minus 1024. You set the maximum number of incoming connections using the Resource Limits page (**Cache-Specific** > **Resource Limits**) of the Oracle9*i*AS Web Cache Manager. You set the cluster member capacity using the Cluster Configuration page (**Administration** > **Cluster Configuration**).

On Windows operating systems, Oracle9*i*AS 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.

Task 9: Specify Settings for Origin Servers

Configure Oracle9*i*AS Web Cache with the application Web servers or proxy servers for which it sends cache misses. Typically, Oracle9*i*AS Web Cache uses application Web servers for internal sites and proxy servers for external sites protected by a firewall.

By default, the listening port and host name of the Oracle HTTP Server are configured. When Oracle9*i*AS Web Cache is installed, Oracle HTTP Server has a default listening HTTP port of 7778 and an HTTPS port of 4444.

Oracle9*i*AS Web Cache will only forward requests to a configured application Web server or proxy server if the server is mapped to a Web site in the Site to Server Mapping page (**General Configuration** > **Site to Server Mapping**). If you are configuring **load balancing** for a site, then create *one* site-to-server mapping that maps *all* the applicable origin servers to the site.

See Also:

"Load Balancing" on page 1-21 for an overview of load balancing

Task 10: Configure Web Site Settings on page 6-26 for instructions for configuring site-to-server mappings

To configure Oracle9*i*AS Web Cache with application Web server or proxy server information:

1. In the navigator pane, select **General Configuration** > **Application Web Servers** or **Proxy Servers**.

   The Application Web Servers or Proxy Servers page appears.

2. In the Application Web Servers or Proxy Servers page, choose **Add**.

   The Create Application Web Server or Create Proxy Server dialog box appears.

3. In the **Hostname** field, enter the host name of the origin server.

4. In the **Port** field, enter the listening port from which the origin server will receive Oracle9*i*AS Web Cache requests.

5. In the **Capacity** field, enter the maximum number of concurrent connections that the origin server can accept.

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

   In a cache cluster, Oracle9*i*AS Web Cache ensures that the total number of connections from all cluster members to the origin server does not exceed the

capacity. Each cluster member is allowed a percentage of the maximum connections, using the following formula:

```
connections_from_each_cluster_member = capacity / number_of_cluster_members
```

6. In the **Failover Threshold** field, enter the number of allowed continuous request failures before Oracle9*i*AS Web Cache considers the origin server down and performs automatic **failover** of the origin servers.

   The default is five requests.

   If a server fails any time after Oracle9*i*AS Web Cache has started to send a request, then Oracle9*i*AS Web Cache increments the failure counter. The failure counter is reset in the event of a successful 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 either `500 Internal Server Error`, `502 Bad Gateway`, `503 Service Unavailable`, or `504 Gateway Timeout` messages

   Once the threshold is met, Oracle9*i*AS Web Cache considers the server down and uses other servers for future requests. When a server is down, Oracle9*i*AS Web Cache starts polling it. It does this by sending requests to the URL specified in the **Ping URL** field. When Oracle9*i*AS Web Cache is able to successfully get a response from the server without any network errors and the HTTP response code is not less than 100, or equal to 500, 502, 503, 504, then Oracle9*i*AS Web Cache considers the server up again and uses it for future requests.

   Note: The threshold does not apply if Oracle9*i*AS Web Cache cannot connect to a server. In this case, Oracle9*i*AS Web Cache immediately considers the server down and does not use it for future requests. The failover to another server does not apply if there is only server left.

7. In the **Ping URL** field, enter the URL that Oracle9*i*AS Web Cache will use to poll an origin server that has reached its failover threshold:

   - For an application Web Server, enter either a relative or fully-qualified URL that includes the domain name, or site name, representing the virtual host of the application Web server

- For a proxy server, you must enter a fully-qualified URL that includes the domain name, or site name, representing the virtual host of the origin server behind the proxy server

Rather than using a static URL, Oracle Corporation recommends using a URL that checks the health of the application logic on the origin server and returns the appropriate HTTP 200 or 500 status codes.

8. In the **Ping Interval (seconds)** field, enter the time in seconds that Oracle9*i*AS Web Cache will poll an origin server that has reached its failover threshold.

   The default is 10 seconds.

9. From the **Protocol** list, select either **HTTP** to send HTTP requests on the port or **HTTPS** to send HTTPS requests on the port.

   See Also: "Secure Sockets Layer (SSL) Support" on page 1-27

10. Choose **Submit**.

11. If you selected HTTPS as the listening protocol, then specify the location of the **wallet** for Oracle9*i*AS Web Cache communication to the application Web server or proxy server.

    This wallet manages Oracle9*i*AS Web Cache authentication data such as keys, certificates, and trusted certificates needed by the **Secure Sockets Layer (SSL)**. By default, wallets are stored in the following locations:

    - `/etc/ORACLE/WALLETS/`*`user_name`* on UNIX
    - `%USERPROFILE%\ORACLE\WALLETS` on Windows operating systems

    Oracle Corporation recommends entering the location, even if the default is being used.

    To specify the wallet location:

    a. In the navigator pane, select **Cache-Specific Configuration** > **Origin Server Wallet**.

       The Origin Server Wallet page appears.

    b. In the Origin Server Wallet page, select the cache for which to modify wallet settings, and then choose **Edit**.

       The Edit Origin Server Wallet dialog box appears.

    c. In the Wallet Directory field, enter the location of the wallet in the **Wallet** field.

d. Choose **Submit**.

See Also: "Configuring Oracle9iAS Web Cache for HTTPS Requests" on page 6-38 for complete instructions for HTTPS configurations

Task 10: Configure Web Site Settings

For Oracle9*i*AS Web Cache to act as a virtual server for one or more Web sites, configure Oracle9*i*AS Web Cache with information about the Web site.

To configure settings for a Web site:

1. Specify a site definition:

Notes:

- It may not be possible to specify a site definition for an external **ESI provider site**. If an ESI request is made to a provider that does not match any application Web server mapping, then Oracle9*i*AS Web Cache uses Domain Name System (DNS) to resolve the site name. Note that this will not work if there is a firewall between the cache and the ESI provider. In that case, you must provide a proxy server mapping that directs the request to the appropriate proxy.

  Undefined ESI provider sites disable the following Oracle9*i*AS Web Cache features:

  — Performance assurance heuristics

  — Origin server features, such as surge protection, load balancing, failover, and **session binding**
- It is not possible to configure only ESI provider sites. In a configuration with ESI provider sites, at least one **virtual host site** definition must exist for ESI template pages.

---

a. In the navigator pane, select **General Configuration** > **Sites**.

The Site Definitions page appears.

b. In the Site Definitions page, choose **Add Site**.

The Add Site dialog box appears.

c. In the **Host Name** field, enter the site name, such as `www.company.com`.

Note: Do not use the wildcard * in the **Host Name** field to represent multiple sites.

d. In the **Port Number** field, enter the port number from which the Web site is listening for incoming HTTP requests.

The port number should be the port used in browser requests.

e. In the **HTTPS Only Prefix** field, enter the URL prefix for which only HTTPS requests will be served. If all traffic must be restricted to HTTPS, enter "/ " for the entire site.

f. In the **Default Site** field, select **Yes** to specify the sites as the default site, or select **No** to specify this site as a nondefault site.

If you select **Yes** for a site, another site that previously had the **Yes** setting will change to **No**.

See Also: "Default Site Settings" on page 6-33 for information about how the default site is used

g. In the **Create Alias from Site Name with/without www** field, select either **Yes** or **No**.

Many sites are represented by one or more aliases. Oracle9*i*AS Web Cache recognizes and caches requests for a site and its aliases. For example, site `www.company.com:80` may have an alias of `company.com:80`. By specifying this alias, Oracle9*i*AS Web Cache caches the same content from either `company.com:80` or `www.company.com:80`. If a request includes a site alias that is not configured, then Oracle9*i*AS Web Cache sends the request to the default site.

– Select **Yes** to use the site name as a site alias.

  For example, if the site domain name is `company.com`, a site alias of `www.company.com` will be used. If the site domain name is `www.company.com`, a site alias of `company.com` will be used.

– Select **No** if you do not want to use the site name as a site alias.

If additional or other site aliases are used by the site, continue to Step 2. Otherwise, skip to Step 3.

h. Choose **Submit**.

2. If the site uses additional aliases, map the site to those aliases.

   Important: To ensure requests are directed to the correct site, specify all possible variations of the site name. If a request includes a site alias that is not configured, then Oracle9*i*AS Web Cache sends the request to the default site.

   a. In the Site Definitions page, select a site, and then choose **Add Alias**.

      The Add Alias for Site dialog box appears.

   b. In the **Host Name** field, enter the site alias name, such as `company.com`.

      Note: Do not use the wildcard * in the **Host Name** field to represent multiple aliases.

   c. In the **Port Number** field, enter the HTTP or HTTPS port number from which the alias is listening for incoming HTTP requests.

      The port number should be the port used in browser requests.

3. Map the site to the origin servers:

   a. In the navigator pane, select **General Configuration** > **Site to Server Mapping**.

      The Site to Server Mapping page appears.

   b. In the Site to Server Mapping page, choose **Create** if no mappings exist. If mappings already exist, select a mapping, and then choose **Insert Above** or **Insert Below**.

      The Create Site to Server Mapping or Edit / Add Site to Server Mapping dialog box appears.

   c. In the **Edit Site Name** section, select one of the following options:

      – **Enter Site Name** to enter the site name, such as `www.company.com` or `*.company.com`, as well as the HTTP or HTTPS port number from which the site is listening for incoming requests.

      – **Select from Site definitions** to select a site definition created in the Site Definitions page.

Notes: You can use the wildcard * in the **Host Name** field in the following ways:

- Map multiple site names to one or more application Web server or proxy servers. For example, `*.company.com` can be used to match sites `site1.company.com` and `site2.company.com`.
- Route cache misses to undefined ESI provider sites protected by a firewall and accessible by a proxy server. For example, * can be used to map to proxy server `proxy-host`.

You can use the wildcard * in the **Port Number** field to map the same site name to different port numbers with the same origin servers.

This option does not enable you to create a site definition. You must create a site definition in the Site Definitions page.

d. In the **Select Origins Servers to which this Site is mapped**, select one of the following options:

– **Select Application Web Servers** to select application Web servers specified in the Application Web Servers page

– **Select Proxy Servers** to select proxy servers specified in the Proxy Servers page

Note: If you configured multiple origin servers in "Task 9: Specify Settings for Origin Servers" on page 6-23 for load balancing, then create *one* site-to-server mapping that maps *all* the applicable origin servers to the site. In that site-to-server mapping, *select all* the origin servers that apply for the site. If you split the origin servers among multiple site-to-server mappings, then load balancing for the site will not occur in the intended manner.

e. In the **Exclude** section, select one of the following options to restrict Oracle9*i*AS Web Cache access to the origin servers for the sites specified in **Edit Site Name**.

– **ESI** restricts Oracle9*i*AS Web Cache from using this mapping for ESI includes. Select this option if the site is a virtual host site that does not provide ESI content.

- **NON_ESI** restricts Oracle9*i*AS Web Cache from using this mapping for any content that is not ESI. Select this option if the site is an ESI provider site.
- **NONE** does not enforce any Oracle9*i*AS Web Cache restrictions. Select this option if the site is a virtual host site that supports ESI.

For example, one mapping entry that uses **Exclude ESI** does not mean that Oracle9*i*AS Web Cache is not allowed to assemble ESI content from other origin servers.

f. Choose **Submit**.

The Edit / Add Site to Server Mapping dialog box closes.

g. In the Site to Server Mapping page, select a mapping, and then choose **Move Up** or **Move Down** to order the mappings. Note the following:

- Higher priority mappings are processed first.
- Because mappings that use the wildcard * encompass a broader scope, give these rules a lower priority than other mappings.

Note: If the protocol used in the `src` attribute of an `<esi:include>` tag attribute does not match the protocol specified in the Site to Server Mapping page, then Oracle9*i*AS Web Cache uses the protocol configured for the origin server in the Site to Server Mapping page. Oracle9*i*AS Web Cache also reports the following warning message to the event log:

Date `Warning: ESI Include protocol does not match Origin Server protocol: Origin Server Protocol=`*protocol* `URL=`*URL*

For example, if the template page is configured with `<esi:include src="https://www.company.com/gifs/frag1.gif"/>` and the Site to Server mapping specifies HTTP for the origin server, then `http://www.company.com/gifs/frag1.gif` is used and the following message appears in the event log:

```
11/Jan/2002:19:25:59 +0000 Warning: ESI Include protocol does
not match Origin Server protocol: Origin Server Protocol=http
URL=https://www.company.com/gifs/frag1.gif
```

---

4. For configured sites, specify apology pages to be served from Oracle9*i*AS Web Cache for network communication errors, site busy errors, and ESI `<esi:include>` errors:

   a. In `$ORACLE_HOME/webcache/docs` on UNIX and *`ORACLE_HOME`*`\webcache\docs` on Windows create apology pages.

      The *default page* setting is applied to **Network error**, **Site busy**, and **Partial page error** apology pages for defined sites:

      – For **Network error**, the *default page* setting is set to `network_error.html`. This apology page is served when there is a network problem while connecting, sending, or receiving a response from an origin server for a cache miss request.

      – For **Site busy**, the *default page* setting is set to `busy_error.html`. This page is server when origin server capacity has been reached.

      – For **Partial page error**, the *default page* setting is not set to any value. Because ESI has its own language elements for exceptions, there is no default for `<esi:include>` errors. If you plan to use `<esi:include>` tags for partial page caching and you do not implement the `onerror` attribute or the `try|attempt|except` block, then you must create an apology page. The `onerror` attribute is used before the `try|attempt|except` block. If the `try|attempt|except` block does not exist, then the exception handling is propagated to the parent or template page. The parent page will use the apology page, `onerror` attribute, or `try|attempt|except` block to handle the error. When an apology page is configured, Oracle9*i*AS Web Cache bypasses any ESI programmatic exception handling code.

      For a production environment, Oracle Corporation advises that you modify the defaults or create entirely new apology pages to be consistent with other error pages for the site. You can modify the settings for apology pages in one of two ways:

      – Change the values of the **Default Apology Pages** rule, and apply it to all defined sites.

      – Modify the apology page settings for a specific site.

   b. In the navigator pane, select **General Configuration** > **Apology Pages**.

      The Apology Pages page appears.

c. Select either ***Default Apology Pages*** or a site name in the table, and then choose **Edit**.

   The Edit Apology Pages dialog box appears.

d. In the **Apology page for network error** field, enter the file name of the apology page that will be delivered for network communication problems between Oracle9*i*AS Web Cache and the Web site.

   If you are using the default `network_error.html` page, then leave the field as is.

e. In the **Apology page for site busy** field, enter the file name of the apology page that will be delivered when a Web site is saturated with requests.

   If you are using the default `busy_error.html` page, then leave the field as is.

f. In the **Apology page for partial page error** field, enter the file name of apology page that will be delivered when Oracle9*i*AS Web Cache is unable to retrieve an HTML fragment for an `<esi:include>` tag.

   If you are not using `<esi:include>` tags for **partial page caching** or you want to use ESI language elements for exceptions, then do not enter a value.

g. Choose **Submit**.

   The Edit Apology Pages dialog box closes.

   If you selected ***Default Apology Pages*** in Step 4c, then the new settings will be applied to all defined sites with the *default page* setting. However, the new setting will not be applied to undefined sites. If you selected a specific site in Step 4c, then the new settings will be applied to just to the site.

   See Also: "Exceptions and Errors" on page D-11 to understand how exceptions and error are handled for `<esi:include>` errors

Default Site Settings

For those requests that do not include a `Host` request-header field, Oracle9*i*AS Web Cache uses the default site settings to determine the appropriate site for the requests. Figure 6–1 shows the default site definition and site-to-server mappings.

The default site established during installation uses the host name and listening port of the computer on which the Oracle9*i* Application Web Server was installed. The default site-to-server mappings use the following rules:

- The first rule maps HTTP requests to the Oracle HTTP Server. The **Exclude NONE** setting enables Oracle9*i*AS Web Cache to serve site content, as well as assemble ESI include fragments.
- The second rule maps HTTPS requests to the Oracle HTTP Server
- The third rule uses a * wildcard host name to map all other virtual site names to the Oracle HTTP Server and a * wildcard port number to map the site name to multiple port numbers. The **Exclude ESI** setting restricts Oracle9*i*AS Web Cache from fetching ESI content from any sites other than the sites specified in the first two rules.

Figure 6–1 Default Site Settings

Site Definitions

Select	Site				Select	Aliases	
	Host Name	Port	HTTPS Only Prefix	Default		Host Name	Port
○	host-server	7777		Yes	N/A	None	

Edit Site | Add Site | Delete Site | Edit Alias | Add Alias | Delete Alias

Site to Server Mapping

Select	Priority	Site			Origin Server		
		Host Name	Port	Exclude	Host Name	Port	Proxy
○	1	host-server	7777	NONE	host-server	7778	No
○	2	host-server	4443	NONE	host-server	4444	No
○	3	*	*	ESI	host-server	7778	No

Edit Selected... | Delete Selected... | Insert Above... | Insert Below...
Move Up | Move Down

Virtual Host Site Example Settings

A virtual host site named `www.company.com` without ESI content could have the following site definition and site-to-server mapping shown in Figure 6–2.

The site definition specifies `www.company.com`, port 80 as the site and `company.com`, port 80 as the site alias. The site-to-server rule maps requests to `www.company.com` to application Web server `host-server`, port 7778. The **Exclude ESI** setting restricts Oracle9*i*AS Web Cache from fetching ESI content from `host-server`, port 7778 for `www.company.com:80`.

Figure 6–2 Example: Site Settings for a Virtual Host Site

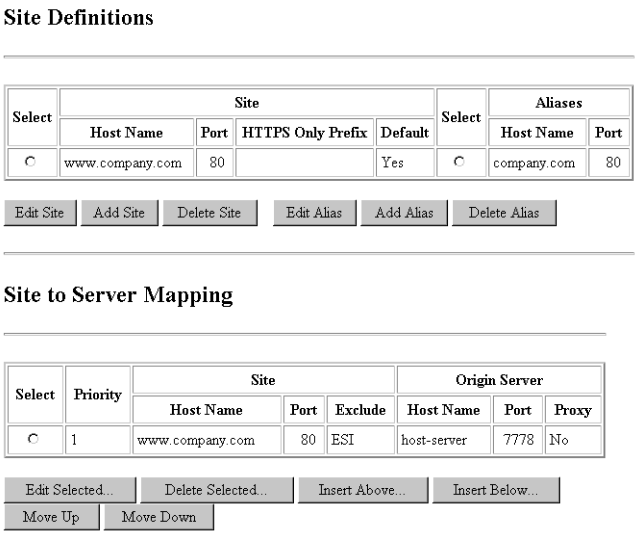


Figure 6–3 shows the site definitions and site-to-server mappings for virtual host sites `www.1st.company.com` and `www.2nd.company.com` that support ESI.

The site definition specifies `www.1st.company.com`, port 80 and `www.2nd.company.com`, port 80 as the sites, and `1st.company.com`, port 80 and `2nd.company.com`, port 80 as the site aliases. The site-to-server rules map sites matching `www.*.company.com` to application Web server `host-server`, port 7778. The **Exclude NONE** setting enables Oracle9*i*AS Web Cache to serve site content, as well as assemble ESI include fragments.

Figure 6–3 Example: Site Settings for Multiple Virtual Host Sites

Site Definitions

Select	Site				Select	Aliases	
	Host Name	Port	HTTPS Only Prefix	Default		Host Name	Port
○	www.1st.company.com	80		Yes	○	1st.company.com	80
○	www.2nd.company.com	80		No	○	2nd.company.com	80

Edit Site | Add Site | Delete Site | Edit Alias | Add Alias | Delete Alias

Site to Server Mapping

Select	Priority	Site			Origin Server		
		Host Name	Port	Exclude	Host Name	Port	Proxy
○	1	www.*.company.com	80	NONE	host-server	7778	No

Edit Selected... | Delete Selected... | Insert Above... | Insert Below...
Move Up | Move Down

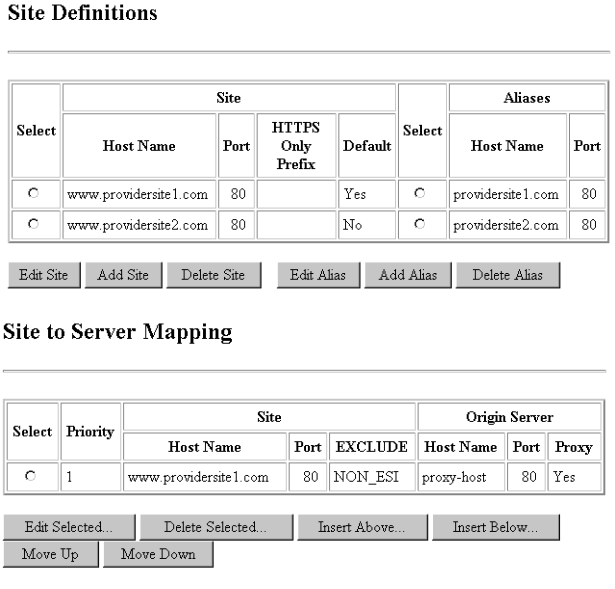
ESI Provider Site Example Settings

ESI provider sites named `www.providersite1.com` and `www.providersite2.com` could have the following site definition and site-to-server mapping shown in Figure 6–4.

The site definition specifies `www.providersite1.com`, port 80 and `www.providersite2.com`, port 80 as the sites, and `providersite1.com`, port 80 and `providersite2.com`, port 80 as the site aliases. The site-to-server rules maps `www.providersite1.com` to proxy server `proxy-host`, port 80. The **Exclude NON_ESI** setting restricts Oracle9*i*AS Web Cache from using this mapping for any content which is not ESI. There is no mapping for `www.providersite2.com`, because the proxy server is not known. Instead, DNS will be used to resolve the site name to the appropriate server. In addition, other ESI provider sites that do not have site definitions will also be resolved through DNS.

Note: This example only shows ESI provider site mappings. In an actual deployment, at least one virtual host definition must exist for ESI template pages.

Figure 6–4 Example: Site Settings for Multiple ESI Provider Sites



Site Definitions

Select	Site				Select	Aliases	
	Host Name	Port	HTTPS Only Prefix	Default		Host Name	Port
○	www.providersite1.com	80		Yes	○	providersite1.com	80
○	www.providersite2.com	80		No	○	providersite2.com	80

Edit Site | Add Site | Delete Site | Edit Alias | Add Alias | Delete Alias

Site to Server Mapping

Select	Priority	Site			Origin Server		
		Host Name	Port	EXCLUDE	Host Name	Port	Proxy
○	1	www.providersite1.com	80	NON_ESI	proxy-host	80	Yes

Edit Selected... | Delete Selected... | Insert Above... | Insert Below...
Move Up | Move Down

Task 11: Specify Caching Rules

Specify the URLs containing the documents you want Oracle9*i*AS Web Cache to cache.

See Also: "Configuring Caching Rules" on page 7-8

Task 12: Apply Changes and Restart Oracle9*i*AS Web Cache

After Oracle9*i*AS Web Cache is configured, apply changes and restart Oracle9*i*AS Web Cache.

To apply changes, in the Oracle9*i*AS Web Cache Manager main window, choose **Apply Changes**.

To restart Oracle9*i*AS Web Cache, use either Oracle9*i*AS Web Cache Manager or the `webcachectl` utility on the computer on which Oracle9*i*AS Web Cache software is installed and configured.

Use Oracle9*i*AS Web Cache Manager...	Use the webcachectl Utility...
1. Start Oracle9*i*AS Web Cache Manager. **See Also:** "Starting Oracle9iAS Web Cache Manager" on page 5-3	From the command line, enter: `webcachectl restart`
2. In the navigator pane, select **Administration** > **Operations**. The Operations page appears.	
3. In the Operations page, choose **Restart.**	

When you restart Oracle9*i*AS Web Cache, all objects are cleared from the cache. In addition, all statistics are cleared.

Note: You must restart the `admin` server process with the `webcachectl restart` command rather than with the **Restart** option if you modified either of the following configuration settings:

- `administrator` password, invalidator password, or trusted subnet settings in the Security page (**General Configuration** > **Security**)
- Operations ports in the Operations Ports page (**Administration** > **Operations**)

See Also: "Starting and Stopping Oracle9iAS Web Cache" on page 8-2

Configuring Oracle9*i*AS Web Cache for HTTPS Requests

You can configure Oracle9*i*AS Web Cache to receive HTTPS browser requests and send HTTPS requests to the origin server. HTTPS uses the **Secure Sockets Layer (SSL)** to encrypt and decrypt user page requests as well as the pages that are returned by the origin server.

To configure HTTPS support, perform these tasks:

- Task 1: Create Wallets
- Task 2: Configure HTTPS Ports and Wallet Location
- Task 3: (Optional) Permit Only HTTPS Requests for a Site

Task 1: Create Wallets

Wallets are needed to support the following HTTPS requests:

- Browser requests for sites hosted by Oracle9*i*AS Web Cache
- Administration, invalidation, and statistics monitoring requests to Oracle9*i*AS Web Cache
- Oracle9*i*AS Web Cache requests to origin servers

Each site requires at least one wallet. One wallet can be shared among all the Oracle9*i*AS Web Cache listening ports, or a separate wallet can be created for each Oracle9*i*AS Web Cache listening port.

To create a wallet, use Oracle Wallet Manager. Create the wallet as the following user:

- The user and group ID configured in the Process Identity page (**Cache-Specific Configuration** > **Process Identity**) on UNIX
- The owner of `Oracle`*`HOME_NAME`*`WebCache` service on Windows operating systems

When the `webcachectl` or `Oracle`*`HOME_NAME`*`WebCache` service starts the `cache` server process, Oracle9*i*AS Web Cache opens the wallet as the `webcachectl` or the `Oracle`*`HOME_NAME`*`WebCache` service owner.

By default, wallets are stored in the following locations:

- `/etc/ORACLE/WALLETS/`*`user_name`* on UNIX
- `%USERPROFILE%\ORACLE\WALLETS` on Windows operating systems

See Also:

- "Secure Sockets Layer (SSL) Support" on page 1-27
- *Oracle9i Application Server Security Guide*

Enabling Wallets to Open on Windows

Oracle9*i*AS Web Cache attempts to open wallets at startup on Windows. On Windows, wallets are protected so that only the user that created them can open and use them. By default, Oracle9*i*AS Web Cache services are associated with the local system account, which does not have permission to open wallets.

To enable Oracle9*i*AS Web Cache to open wallets at startup:

1. Create a wallet with an administrator account.
2. Change the system account information for the Oracle9*i*AS Web Cache services:

Windows NT

1. Choose the **Services** icon from the Control Panel window.

   The Services window appears.
2. Select the `Oracle`*`HOME_NAME`*`WebCache` service.

   The Service dialog appears.
3. Choose **This Account**.

   By default the LocalSystem user account is associated with the service.
4. Choose the ellipse (**...**) 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 - 8 for the `Oracle`*`HOME_NAME`*`WebCacheAdmin` and `Oracle`*`HOME_NAME`*`WebCacheMon` service.
10. In the Services window, choose **Close**.

Windows 2000

1. Choose **Administrative Tools** > **Services** from the Control Panel window.

   The Services window appears.
2. Select the `Oracle`*`HOME_NAME`*`WebCache` service.

   The Oracle*HOME_NAME*WebCache 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 Oracle*HOME_NAME*WebCache 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 `Oracle`*`HOME_NAME`*`WebCacheAdmin` and `Oracle`*`HOME_NAME`*`WebCacheMon` services.

On Windows NT, additionally grant the wallet administrator the right to run Oracle9*i*AS Web Cache as a service:

1. Choose **Start** > **Programs** > **Administrative Tools** > **User Manager**.

   The User Manager window appears.
2. Select the wallet administration, and then choose **Policies** > **User Rights**.

The User Rights Policy dialog box appears.

3. Choose the **Show Advanced User Rights** check box, and then select **Log on as a service** from the **Right** list.
4. Select **Users** from the **Grant To** list.

   If **Users** does not exist, create it:

   a. Choose **Add**.

      The Add Users and Groups dialog box appears:

   b. Select the name of the local host computer from the **List Names From** list.

   c. Select **Users** from the **Names** list, and then choose **Add**.

   d. Choose **OK**.

      Users appears in the **Grant To** list.
5. Choose **OK** in the User Rights Policy dialog box.

   The User Manager window reappears.
6. Choose **User** > **Exit**.

See Also: "Wallet Cannot Be Opened" on page 10-6

Task 2: Configure HTTPS Ports and Wallet Location

To configure HTTPS protocol support between browsers and Oracle9*i*AS Web Cache:

1. Select **Cache-Specific Configuration** > **Listening Ports** in Oracle9*i*AS Web Cache Manager to configure Oracle9*i*AS Web Cache with an HTTPS listening port and the location of the wallet for each supported site.
2. Select **Cache-Specific Configuration** > **Operations Ports** in Oracle9*i*AS Web Cache Manager to configure administration, invalidation, and statistics monitoring requests with HTTPS listening ports and the location of the site's wallet.

   The ports for these requests can share the same wallet as established for the Oracle9*i*AS Web Cache listening port in Step 1.

   See Also:

   - "Task 3: Configure Oracle9iAS Web Cache with Listening Ports for Incoming Browser Requests" on page 6-6
   - "Task 5: Configure Oracle9iAS Web Cache with Operations Ports" on page 6-9

To configure HTTPS protocol support between Oracle9*i*AS Web Cache and origin servers:

1. Select **General Configuration** > **Application Web Servers** or **Proxy Servers** in Oracle9*i*AS Web Cache Manager to configure an application Web server or proxy server with an HTTPS communication port.
2. Select **Cache-Specific Configuration** > **Origin Server Wallet** in Oracle9*i*AS Web Cache Manager to specify the location of the wallet used for communication from Oracle9*i*AS Web Cache to an origin server.

   The ports for these requests can share the same wallet as established for the Oracle9*i*AS Web Cache listening ports.

See Also:

- "Task 9: Specify Settings for Origin Servers" on page 6-23 to specify the wallet for the origin server
- "Task 3: Configure Oracle9iAS Web Cache with Listening Ports for Incoming Browser Requests" on page 6-6 to specify the wallet for Oracle9*i*AS Web Cache
- "Task 5: Configure Oracle9iAS Web Cache with Operations Ports" on page 6-9 to specify the wallet for the operations ports

Task 3: (Optional) Permit Only HTTPS Requests for a Site

You can restrict a URL or set of URLs for a site to permit only HTTPS requests.

To allow only HTTPS traffic for a URL or a set of URLs:

1. Configure Web site settings, as described in "Task 10: Configure Web Site Settings" on page 6-26.
2. In Step 1e, enter the URL or URL prefix.

   If all traffic must be restricted to HTTPS, enter "/" for the entire site.

Configuring Multiple Origin Servers

This section describes additional configuration options available for deployments with two or more origin servers.

This section contains these topics:

- Configuring Load Balancing and Failover
- Binding a Session to an Origin Server

Configuring Load Balancing and Failover

For those requests that Oracle9*i*AS Web Cache cannot serve, you can distribute the requests over a set of origin servers with the load balancing feature. To configure load balancing for a site, configure the **capacity** for each origin server, and create *one* site-to-server mapping that maps *all* the applicable origin servers to the site.

When load balancing is configured and an origin server is no longer available, Oracle9*i*AS Web Cache automatically performs failover of the origin servers. Oracle9*i*AS Web Cache knows if an origin server is down when a failover threshold has been met.

An origin server can become unavailable if it is taken down for reconfiguration or there is a network or hardware failure. In these scenarios, Oracle9*i*AS Web Cache automatically distributes the load over the remaining origin servers and polls the failed origin server for its current up or down status until it is back online. Existing requests to the failed origin server result in errors. However, new requests are directed to the other origin servers. When the failed server returns to operation, Oracle9*i*AS Web Cache includes it in the load distribution.

See Also:

- "Load Balancing" on page 1-21 for an overview of load balancing
- "Backend Failover" on page 23 for an overview of failover
- "Task 9: Specify Settings for Origin Servers" on page 6-23 for instructions on specifying capacity and the failover threshold
- "Task 10: Configure Web Site Settings" on page 6-26 for instructions on creating site-to-server mappings

Binding a Session to an Origin Server

> **Note:** If an origin server is busy, then Oracle9*i*AS Web Cache disables session binding to that origin server.

You can configure Oracle9*i*AS Web Cache to support session binding, whereby a user session for a particular site is bound to an origin server in order to maintain state for a period of time. To utilize this feature, the origin 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, Oracle9*i*AS Web Cache can keep track of sessions between Web browsers and origin servers. A session cookie or an embedded URL parameter enables Oracle9*i*AS Web Cache to bind a particular user session to a specific origin server.

> **See Also:** "Session Binding" on page 1-25 for an overview of origin server binding

By default, session binding is not enabled for any sites. If there is no session binding specified for a site, then the *default session binding* setting is applied, which uses the ***Default Session Binding*** rule. The ***Default Session Binding*** rule has a default value of Session Binding Disabled. You can enable session binding in one of two ways:

- Change the default value of the ***Default Session Binding*** rule from Session Binding Disabled to some other session, and apply it to all defined sites.
- Overwrite the *default session binding* setting to some other session for a specific site.

To enable session binding:

1. Start Oracle9*i*AS Web Cache Manager.

   > **See Also:** "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. In the navigator pane, select **General Configuration** > **Session Management** > **Session Binding**.

   The Session Binding page appears.

3. In the Session Binding page, select ***Default Session Binding*** or a specific site name in the table, and then choose **Edit**.

The Change / Add Session Binding dialog box appears.

4. From the **Please select a session** list:

   - If you selected the ***Default Session Binding*** rule in Step 3, change the session value from Use Default Session Binding to another defined session, and then skip to Step 7.
   - If you selected a specific site in Step 3, then change the session value from Disable Session Binding to another defined session, and then skip to Step 7.

   If the sessions listed do not contain the definition you require, then choose **Cancel** to exit the Change / Add Session Binding dialog box. Continue to Step 5.

5. Create a session definition:

   a. In the navigator pane, select **General Configuration** > **Session Management** > **Session/Personalized Definitions**.

      The Session / Personalized Attribute Definitions page appears.

   b. From the **For Site** list, select the Web site for which to create site-specific site definitions.

   c. Choose **Add** or **Create**.

      The Create Session / Personalized Attribute Definitions page appears.

   d. In the **Session/Attribute** field, enter an easy-to-remember unique name for the attribute.

   e. Enter the cookie name in the **Cookie Name** field and 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: Oracle9*i*AS 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 for Oracle9*i*AS 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 the origin 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.

Oracle9*i*AS 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, Oracle9*i*AS Web Cache does not continue to bind the user session to the origin server. Instead, Oracle9*i*AS Web Cache uses load balancing to choose an origin server. To avoid pages being served past the browser session expiration time, ensure that the session cookie expires before the origin server expires the browser session.

f. Choose **Submit**.

g. Repeats Steps 1 through 4.

6. In the **Inactivity Timeout** field, enter the number of minutes you want Oracle9*i*AS Web Cache to wait before timing out an inactive session to the origin server.

Oracle Corporation recommends setting the value to a higher value than the inactivity timeout set for the Web site.

7. Choose **Submit**.

   The Change / Add Session Binding dialog box closes.

   If you selected the ***Default Session Binding*** rule in Step 3, then the new settings will be applied to all defined sites with the *default session binding* setting. However, the new default will not be applied to undefined sites. If you selected a specific site in Step 3, then the new settings will be applied to just the site.

Configuring a Hierarchy of Caches

This section describes additional configuration options available for **cache hierarchy** deployments.

This section contains these topics:

- Configuring a Distributed Cache Hierarchy
- Configuring an ESI Cache Hierarchy

> **See Also:** "Cache Hierarchies" on page 1-16 for an overview of cache hierarchies

Configuring a Distributed Cache Hierarchy

In a **distributed cache hierarchy**, the **central cache** stores content from application Web servers, and the **remote cache** stores content from the central cache. In a distributed cache hierarchy, the central caches acts as origin servers to the remote cache

To configure a distributed cache hierarchy, perform the tasks in "Setting Up Oracle9iAS Web Cache" on page 6-2 for each cache. When performing the tasks, take special care to perform the following:

1. Configure the correct origin server:
   - For the central cache, configure the central origin servers in the Application Web Servers page or Proxy Servers page (**General Configuration** > **Application Web Servers** or **Proxy Servers**).
   - For the remote caches, configure the central cache as the origin server in the Application Web Servers page.

2. Create the same site definition for both the central and remote caches in the Site Definitions page (**General Configuration** > **Sites**).
3. For both central and remote caches, map the site definition to the origin server (configured in Step 1) in the Site to Server Mapping page (**General Configuration** > **Site to Server Mapping**):
   - For the central cache, map the site to the application Web server or proxy server.
   - For the remote cache, map the site to the central cache.

   When content from the central cache becomes invalid, an invalidation message is sent to its cache. In addition, the central cache propagates the invalidation message to the remote caches.

Note: In order for automatic propagation of invalidation messages to work, Oracle9*i*AS Web Cache passes the encoded `invalidator` password in the page request between the remote and central cache. This HTTP traffic is susceptible to network sniffing. If the network is unprotected and insecure, configure HTTPS ports as follows:

1. In the Listening Ports page (**Cache-Specific Configuration** > **Listening Ports**) of the central cache, disable the default HTTP port. An HTTPS port is already configured by default.
2. In the Operations page (**Cache-Specific Configuration** > **Operations Ports**) of the remote cache, disable the default HTTP port and configure an HTTPS port in its place.

Table 6–1 shows the example settings for the deployment depicted in "Deploying a Distributed Cache Hierarchy" on page 4-29.

Table 6–1 Settings for us.webche-host and jp.webche-host

Setting Location in Oracle9*i*AS Web Cache Manager	Central Cache us.webche-host	Remote Cache jp.webche-host
Listening Port (**Cache-Specific Configuration** > **Listening Ports**)	Port Number: `7777`	Port Number: `7777`
Application Web Server (**General Configuration** > **Application Web Servers** or **Proxy Servers**)	Host Name: `us.server1-host` Port Number: `7778` Host Name: `us.server2-host` Port Number: `7778`	Host Name: `us.webche-host` Port Number: `7777`
Site Definition (**General Configuration** > **Sites**)	Host Name: `www.server.com` Port Number: `80`	Host Name: `www.server.com` Port Number: `80`
Site-to-Server Mapping (**General Configuration** > **Site to Server Mapping**)	Site Host Name and Port: `www.server.com:80` Origin Server Host Name and Port: `us.server1-host:7778` `us.server2-host:7778`	Site Host Name and Port: `www.server.com:80` Origin Server Host Name and Port: `us.webche-host:7777`

Configuring an ESI Cache Hierarchy

In an **ESI cache hierarchy**, a **provider cache** stores content from an **ESI provider site**, and a **subscriber cache** stores content from the origin servers for a local site and contacts provider caches for ESI fragments. In an ESI cache hierarchy, the provider caches acts as origin servers to the subscriber cache.

To configure an ESI cache hierarchy, perform the tasks in "Setting Up Oracle9iAS Web Cache" on page 6-2 for each cache. When performing the tasks, take special care to perform the following:

1. Configure the correct origin server:
   - For each provider cache, configure the origin servers of the ESI provider site in the Application Web Servers page or Proxy Servers page (**General Configuration** > **Application Web Servers** or **Proxy Servers**)
   - For the subscriber cache, configure the origin servers of the local site and the provider caches in the Application Web Servers page.
2. Create site definitions:
   - For each provider cache, create a site definition for the ESI provider site in the Site Definitions page (**General Configuration** > **Sites**).
   - For the subscriber cache, create site definitions for the local site and each ESI provider site in the Site Definitions page.

Note: It may not be possible to specify a site definition for all external ESI provider sites. If an ESI request is made to a provider that does not match any application Web server mapping, then Oracle9*i*AS Web Cache uses DNS to resolve the site name. Note that this will not work if there is a firewall between the cache and the ESI provider. In that case, you must provide a proxy server mapping that directs the request to the appropriate proxy.

---

3. For both subscriber and provider caches, map the site definition to the origin server (configured in Step 1) in the Site to Server Mapping page (**General Configuration** > **Site to Server Mapping**):

   - For the provider caches, map the site definition to the origin server of the ESI provider site.
   - For the subscriber cache, map the local site definition to the origin server for that site, and map each ESI provider site definition to its respective provider cache

   When content from the provider cache becomes invalid, an invalidation message is sent to its cache. In addition, the provider cache propagates the invalidation message to the subscriber cache.

   Note: In order for automatic propagation of invalidation messages to work, Oracle9*i*AS Web Cache passes the encoded `invalidator` password in the page request between the subscriber and provider cache. This HTTP traffic is susceptible to network sniffing. If the network is unprotected and insecure, configure HTTPS ports as follows:

   1. In the Listening Ports page (**Cache-Specific Configuration** > **Listening Ports**) of the provider cache, disable the default HTTP port. An HTTPS port is already configured by default.
   2. In the Operations page (**Cache-Specific Configuration** > **Operations Ports**) of the subscriber cache, disable the default HTTP port and configure an HTTPS port in its place.

Table 6–2 shows the example settings for the deployment depicted in "Multiple Internal ESI Provider Sites" on page 4-18.

Table 6–2 Settings for webche1 and webche2

Setting Location in Oracle9*i*AS Web Cache Manager	Subscriber Cache webche-host	Provider Cache webche-providerhost
Listening Port (**Cache-Specific Configuration** > **Listening Ports**)	Port Number: `7777`	Port Number: `7777`
Application Web Server (**General Configuration** > **Application Web Servers**)	Host Name: `server-host` Port Number: `7778` Host Name: `provider1-host` Port Number: `7778` Host Name: `webche-providerhost` Port Number: `7777`	Host Name: `provider2-host` Port Number: `7778`
Site Definition (**General Configuration** > **Sites**)	Host Name: `www.server.com` Port Number: `80` Host Name: `www.providersite1.com` Port Number: `80` Host Name: `www.providersite2.com` Port Number: `80`	Host Name: `www.providersite2.com` Port Number: `80`

Table 6–2 (Cont.) Settings for webche1 and webche2

Setting Location in Oracle9*i*AS Web Cache Manager	Subscriber Cache webche-host	Provider Cache webche-providerhost
Site-to-Server Mapping (**General Configuration** > **Site to Server Mapping**)	Site Host Name and Port: `www.server.com:80` Origin Server Host Name and Port: `server-host:7778`	Site Host Name and Port: `www.providersite2.com:80` Origin Server Host Name and Port: `provider2-host:7778`
	Site Host Name and Port: `www.providersite1.com:80` Origin Server Host Name and Port: `provider1-host:7778`	
	Site Host Name and Port: `www.providersite2.com:80` Origin Server Host Name and Port: `webche-providerhost:7777`	

Configuring a Cache Cluster

To increase the availability and scalability of your Web site, you can configure multiple instances of Oracle9*i*AS Web Cache to run as members of a cache cluster.

To configure a cache cluster, you specify the general cluster information in the Cluster Configuration page of the Oracle9*i*AS Web Cache Manager and add two or more Oracle9*i*AS Web Cache instances to the cache cluster.

A cache cluster uses one configuration that is propagated from the current cache (the cache to which your browser is connected) to all cluster members. The configuration contains settings that are the same for all cluster members as well as cache-specific settings for each cluster member.

The following settings pertain to all members of a cluster:

- Security
- Sites
- Apology pages
- Auto-restart
- Site to origin server mappings
- Application Web servers
- Proxy servers
- Cacheability and expiration rules
- Session management

The following settings are specific to each member of the cluster:

- Process identity
- Operations ports, such as administration, invalidation, and statistics ports
- Listener ports
- Resource limits
- Event logs
- Access logs
- Origin server wallet

Because a cache cluster contains two or more instances of Oracle9*i*AS Web Cache, you must have two or more instances of Oracle9*i*AS Web Cache installed on one or more nodes before you configure a cache cluster. The instances must be the same version of Oracle9*i*AS Web Cache.

To configure a cache cluster, perform these tasks:

- Task 1: Configure the Cache Cluster Settings
- Task 2: Add Caches to the Cluster
- Task 3: Propagate the Configuration to Cluster Members

> **See Also:** Chapter 3, "Cache Clustering" for an overview of cache clusters

Task 1: Configure the Cache Cluster Settings

To configure the settings for a cache cluster:

1. In the navigation pane, select **Administration > Cluster Configuration.**

   The Cluster Configuration page appears. The General Cluster Information section displays the default clusterwide values for failover and invalidation propagation. The Cluster Members table displays the current cache (the cache to which you are connected) as the only cluster member. Oracle9*i*AS Web Cache ignores the cluster information if there is only one cluster member.

2. In the **General Cluster Information** section of the Cluster Configuration page, choose **Edit.**

   The **Change General Cluster Information** dialog box appears.

3. In the **Cluster Name** field, enter a name for the cluster.

4. In the **Failover Threshold** field, enter the number of allowed consecutive request failures before Oracle9*i*AS Web Cache considers another cache cluster member to have failed.

   Oracle9*i*AS Web Cache considers a request to another cache cluster member to have failed if:

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

   For each failed request, Oracle9*i*AS Web Cache increments the failure counter for that cluster member. This counter is kept separately by each cluster member. When a request is successfully processed by a cluster member, Oracle9*i*AS Web Cache resets the failure counter.

   When the failover threshold is met, Oracle9*i*AS Web Cache considers the cache cluster member to have failed. Oracle9*i*AS Web Cache recalculates the relative capacity of the remaining cache cluster members. It then reassigns ownership of cache content.

   When a cache cluster member is down, Oracle9*i*AS Web Cache starts polling the cache cluster member. It does this by sending requests to the URL specified in the **Ping URL** field. When Oracle9*i*AS Web Cache receives a success response from the cache cluster member, it considers that cache cluster member to be up again. It recalculates the relative capacity of the cache cluster members and it reassigns ownership of cache content.

5. In the **Ping URL** field, enter the URL that cache cluster members will use to attempt to contact a cache cluster member that has reached its failover threshold.

   Use a URL that you can guarantee is stored in each cache. The default is "`/`".

6. In the **Ping Interval** field, enter the time, in seconds, between attempts by a cluster member to reach the failed cluster member.

7. In the **Propagate Invalidation** field, select **Yes** or **No** to specify whether or not you want all invalidation requests from any cache cluster member to be propagated to other cache cluster members.

8. Choose **Submit.**

9. In the Cluster Members table of the Cluster Configuration page, default values are displayed for the current cache. Select the cache and choose **Edit Selected.**

   The Edit Cluster Member dialog box appears.

10. In the **Cache Name** field, enter a name for the Oracle9*i*AS Web Cache instance. The name must be unique from the names of other caches in the cache cluster.

11. By default, the **Host Name** field contains the host name of the node on which Oracle9*i*AS Web Cache is installed. Usually, you do not need to modify this field.

12. By default, the **Oracle Home** field contains the file specification for the Oracle home in which Oracle9*i*AS Web Cache is installed. Usually, you do not need to modify this field. Note that the combination of **Host Name** and **Oracle Home** must be unique in a cache cluster.

13. In the **Capacity** field, enter the number of concurrent incoming connections from other cache cluster members that Oracle9*i*AS Web Cache can sustain.

    This field is used in two different ways:

    - As the absolute capacity for the number of concurrent incoming connections to this cache cluster member from all other cache cluster members.

      The connections are used to receive requests for owned content from other cache cluster members. The number of connections are divided among the other cluster members. For example, in a three-cache cluster, if the capacity of Cache_A is 50, Cache_B can open 25 connections to Cache_A and Cache_C can open 25 connections to Cache_A.

      More connections are used when another cache cluster member contains little or no data in its cache, such as when it is initially started, when it recovers from a failure, or after invalidation. During this time, the cluster member sends many of the requests to its peers, the owners of the content. In most cases, these requests are satisfied more quickly than requests to the origin server. Having a higher number of connections increases performance during this time and shortens the time it takes to fully load the cache. After a cache is fully loaded, fewer of the connections are used. There is no overhead for unused connections.

- As the relative capacity of the cache cluster member.

  The capacity of a cache cluster member is weighted against the total capacity of all active cache cluster members. When you set the capacity, Oracle9*i*AS Web Cache assigns a percentage of the ownership array to the cluster member, indicating how much of the cached content will be owned by the cluster member. The percentage is calculated using the following formula:

  ```
  cluster_member_capacity / total_capacity_of_all_active_cluster_members
  ```

  For example, if cache cluster member Cache_A has a capacity of 100 and cache cluster member Cache_B has a capacity of 300, for a total capacity of 400, Cache_A is assigned 25 percent of the ownership array and Cache_B is assigned 75 percent of the ownership array. That means that Cache_A owns 25 percent of the cached content.

  Note that in calculating the relative capacity, Oracle9*i*AS Web Cache considers the capacity of active cluster members; it does not consider the capacity of cluster members that it has determined to have failed.

In most cases, a capacity of 100 is a reasonable number to use as a starting point.

14. Choose **Submit.**

    You now have one cache, the current cache, in the cluster. However, the cluster information is ignored until you have more than one Oracle9*i*AS Web Cache instance in the cluster.

Task 2: Add Caches to the Cluster

Before you can add a cache to the cluster, the following conditions must be met:

- The cache must exist and must be started. See "Task 12: Apply Changes and Restart Oracle9iAS Web Cache" on page 6-37 for information about starting Oracle9*i*AS Web Cache.
- The administrator password of the cache to be added must be the same as the administrator password of the cache to which you are connected. If it is different, you must connect to the cache's admin server and modify the administration password, as described in "Task 2: Modify Security Settings" on page 6-3.

To add another cache to the cluster:

1. In the navigation pane, select **Administration > Cluster Configuration.**

   The Cluster Configuration page appears.

2. In the Cluster Members section of the Cluster Configuration page, choose **Add.**

   The Add Cache to Cluster dialog box appears.

3. In the **Host Name** field, enter the host name of the cache to be added to the cluster.

4. In the **Admin Port** field, enter the administration port for the cache to be added to the cluster.

   The administration port is the listening port for administrative requests.

5. In the **Cache Name** field, enter a name for the cache. The name must be unique from the names of other caches in the cache cluster.

6. In the **Capacity** field, enter the number of concurrent incoming connections from other cache cluster members that Oracle9*i*AS Web Cache can sustain.

   See Also: Step 13 in "Task 1: Configure the Cache Cluster Settings" on page 6-56 for more information about capacity

7. Choose **Submit.**

   The cache is now part of the cluster and is listed in the Cluster Member table.

8. To add more Oracle9*i*AS Web Cache instances to the cache cluster, repeat Steps 1 through 7.

9. When you have completed adding members to the cache cluster, choose **Apply Changes.**

Oracle9*i*AS Web Cache adds the cache-specific information from the new cache cluster members to the cluster configuration.

You can add more Oracle9*i*AS Web Cache instances to the cluster at any time by choosing **Add.** You can modify the settings for a cache cluster member by choosing **Edit Selected.** You can delete a cache cluster member, other than the current cache, by choosing **Delete Selected.**

Task 3: Propagate the Configuration to Cluster Members

When you modify the cluster and choose **Apply Changes,** Oracle9*i*AS Web Cache adds the cache-specific information from the new cache cluster members to the configuration. For those changes to take affect in all cluster members, you must propagate the configuration and restart the cache server process of the cluster members.

To propagate the configuration to new cluster members:

1. In the navigation pane, select **Administration > Operations.**

   The Operations page appears. The **Operation Needed** column indicates the caches to which the configuration should be propagated.

2. To propagate the configuration to all cache cluster members:

   a. Select **All caches** in the **Operate On** field.

   b. Select an **Interval** to stagger the time that operation begins on the caches.

   c. Choose **Propagate**.

   (Alternatively, you can propagate the configuration to one cluster member at a time. Choose **Selected cache** in the **Operate On** field, and then choose **Propagate**.)

   When the operation completes, the **Operation Needed** column in the Operations page indicates the cluster members that need to be restarted.

3. To stop and restart all cluster members:

   a. Select **All caches** in the **Operate On** field.

   b. Select an **Interval** to stagger the time that operation begins on the caches, and then choose **Restart**. (Alternatively, you can restart one cluster member at a time.)

   c. Choose **Selected cache** in the **Operate On** field.

   d. Choose **Restart**.

When the operation completes, the **Operation Needed** column in the Operations page indicates that no operations are needed. The cache cluster is ready to use.

7

Creating Caching Rules

This chapter explains how to configure caching rules.

This chapter contains these topics:

Caching Rules Overview

Using Oracle9*i*AS Web Cache to specify caching rules, you can select to cache or not to cache content for static documents, multiple-version documents, personalized pages, pages that support a **session cookie** or **embedded URL parameter**, and dynamic pages.

This section discusses the following topics:

- Rule Creation
- Rule Syntax
- Default Caching Rules

Rule Creation

When you create caching rules, you create site-specific caching rules that apply to a particular site and global rules that apply to all sites.

Generally, when you assign caching 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 caching 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

`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`.

Note: Site-specific caching rules are given a higher priority than the global rules.

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 caching rules in Oracle9*i*AS 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 caching 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 caching rules are specified, then Oracle9*i*AS 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.

Rule Syntax

Note that caching 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 7–1 shows examples of content to cache and how to enter regular expression syntax for corresponding caching rules for that content.

Table 7–1 Regular Expression Examples

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

Default Caching Rules

When Oracle9*i*AS Web Cache is installed, site-specific and global caching rules are established for the configured default site.

Figure 7–1 displays the default site-specific caching rules.

Figure 7–1 Default Site-Specific Caching Rules

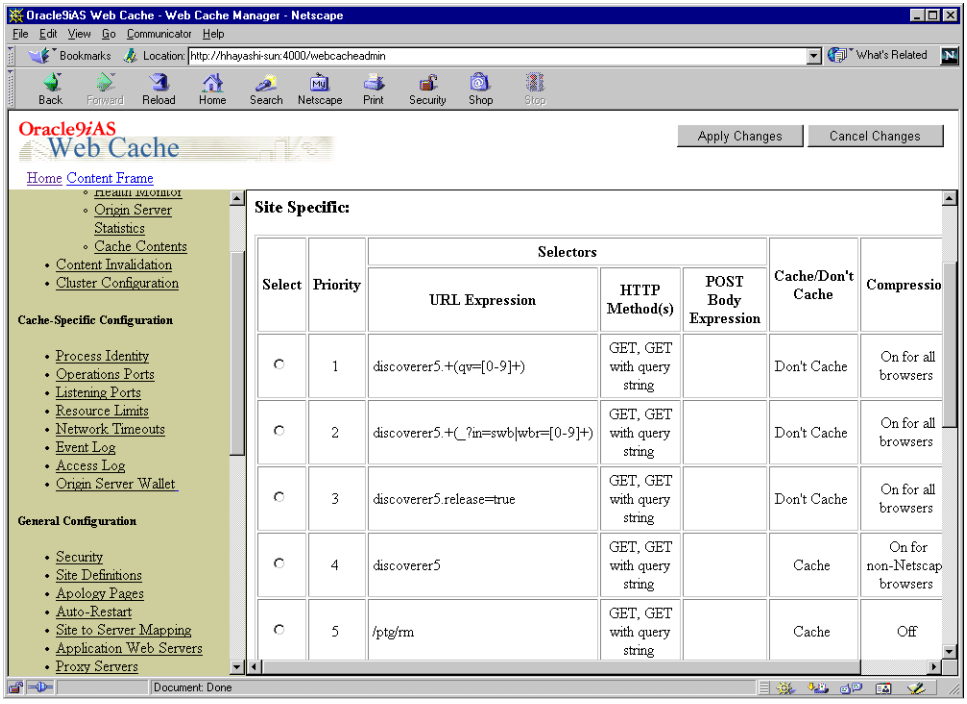


Table 7–2 describes the default site-specific caching rules.

Table 7–2 Default Site-Specific Caching Rules

Priority	URL Expression	HTTP Method(s)	Cache/Don't Cache	Description
1	`discoverer5.(qv=[0-9]+)`	Get, Get with query string	Don't Cache	Instructs Oracle9*i*AS Web Cache to not cache any page that has been generated by the re-run query button
2	`discoverer5.(_ ?in=swblwbr=[0-9]+)`	Get, Get with query string	Don't Cache	Instructs Oracle9*i*AS Web Cache to not cache the Scheduled Workbooks page or workbooks derived from it
3	`discoverer5.release=true`	Get, Get with query string	Don't Cache	Allows consistent execution of Discoverer Plus
4	`discoverer5`	Get, Get with query string	Cache	Instructs Oracle9*i*AS Web Cache to cache documents under the URL `http://`*`host`*`:`*`port`*`/discoverer /discoverer5`
5	`/ptg/rm`	Get, Get with query string	Cache	Instructs Oracle9*i*AS Web Cache cache the default Oracle9*i*AS Wireless servlet. This rule is necessary for Wireless to use Oracle9*i*AS Web Cache to cache transformations. If you change your Wireless servlet mount-point to something other than `/ptg/rm`, update this rule for this to take effect.

Figure 7–2 displays the default global caching rules.

Figure 7–2 Default Global Caching Rules

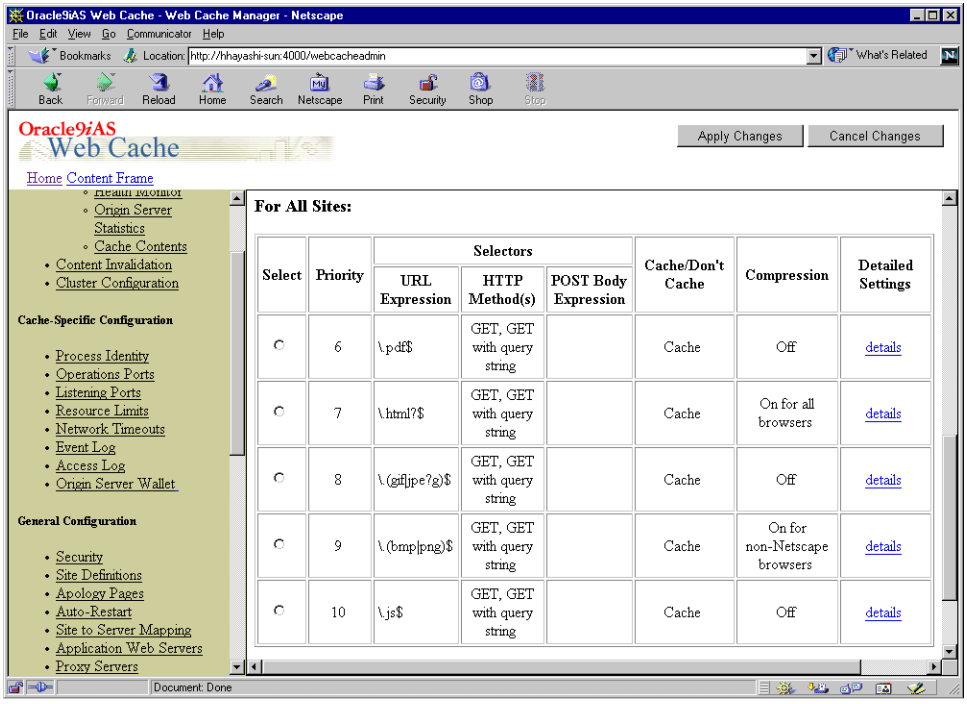


Table 7–3 describes the default global caching rules.

Table 7–3 Default Global Caching Rules

Priority	URL Expression	HTTP Method(s)	Cache/Don't Cache	Description
6	`\.pdf$`	Get, Get with query string	Don't Cache	Instructs Oracle9*i*AS Web Cache to not cache documents ending in `.pdf`
7	`\.html?$`	Get, Get with query string	Cache	Instructs Oracle9*i*AS Web Cache to cache all `.htm` and `.html` files **Note:** HTML pages that contain HTTP authentication response headers are cached. To avoid pages that support basic HTTP authentication from being cached, modify the caching rules to not include pages that require authentication.
8	`\.(gif|jpe?g)$`	Get, Get with query string	Cache	Instructs Oracle9*i*AS Web Cache to cache all `.gif`, `.jpg`, and `.jpeg` files
9	`\.(bmp|png)$`	Get, Get with query string	Cache	Instructs Oracle9*i*AS Web Cache to cache all `.bmp` and `.png` files
10	`\.js$`	Get, Get with query string	Cache	Instructs Oracle9*i*AS Web Cache to cache `.js` (JavaScript) files

Configuring Caching Rules

To configure caching rules:

1. Start Oracle9*i*AS Web Cache Manager.

   See Also: "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. In the navigator pane, select **General Configuration** > **Cacheability Rules**.

   The Cacheability Rules page appears.

3. From the **For Site** list, select the Web site for which to view or create site-specific caching rules.

4. In the Cacheability Rules page, choose **Create Site Specific Rule** or **Create Global Rule** 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.

5. In the **URL Expression** field, enter **regular expression** syntax, matching the URLs to which you want the caching rule to apply.

Remember to use "^" to denote the start of the URL and "$" to denote the end of the URL.

The request URL that browsers send to Oracle9*i*AS Web Cache and the URL that Oracle9*i*AS Web Cache uses internally for that request are different. When Oracle9*i*AS Web Cache serves a page request, it alphabetically sorts any embedded URL parameters of the URL. However, the caching rules are matched against only the internal representation of the URL in which any embedded URL parameters are sorted. To ensure caching rules are matched correctly, sort and enter the embedded URL parameters alphabetically.

For example, consider the URL that follows:

```
http://my.oracle.com/servlet/page?_pageid=53&_dad=moc&_schema=MOC
```

If you enter the regular expression without sorting the embedded URL parameters, `^/servlet/page\?_pageid=53&_dad=moc&_schema=MOC$`, then the caching rule will not match the internal URL used by Oracle9*i*AS Web Cache. To ensure matching, you must enter the regular expression with alphabetically sorted embedded URL parameters, `^/servlet/page\?_dad=moc&_pageid=53&_schema=MOC$`.

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

   You can 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 origin server or database, then do not select **Get with query string** or **POST**. These options should only be selected if the forms are used in search forms.

7. If you selected **POST** in Step 6, 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.

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

9. Optionally, to help track the meaning of rules, enter a comment for the caching rule in the **Comment** field.

   In **ESI Output Permission**, select either **Yes** or **No**. The default is **Yes**.

   - Select **Yes** to enable **Edge Side Includes (ESI)**-compliant proxy caches, such as Akamaii EdgeSuite, to process ESI tags. Select **Yes** only if the following conditions apply:
     - The ESI-compliant cache or service resides between browsers and Oracle9*i*AS Web Cache
     - You prefer the cache or service to perform the ESI processing rather than Oracle9*i*AS 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 7-43

10. Select options for the rows 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

The default is **Yes** and **Compress for all browsers**.

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, Oracle9*i*AS Web Cache does not compress documents containing the following:

  - A `Content-Encoding` response-header field, which is typically used to denote compression

  - A `Content-Disposition` response-header field, which is typically used for attachments

  - Session-encoded URLs, the `<!--WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags, and ESI tags

See Also:

"Compression" on page 1-30 for an overview of compression

"Browser-Specific Issues" on page 10-16 for browser-specific compression issues

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:** ■ "Invalidation and Expiration" on page 2-4 for an overview ■ Step 4 in "Configuring Expiration Rules" on page 7-18 for additional configuration details
Multiple Documents with the Same Selector by Cookies	Select **None** to not have Oracle9*i*AS 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 **General Configuration** > **Cacheability** > **Multiple Documents with the Same Selector** to create a new rule. **See Also:** ■ "Multiple Versions of the Same Document" on page 2-12 for an overview ■ Step 4 in "Configuring Rules for Multiple-Version Documents Containing Cookies" on page 7-21 for additional configuration details

Multiple Documents with the Same Selector by Other Headers

Select the **HTTP request headers** whose values Oracle9*i*AS Web Cache will use to cache and identify multiple-version documents. Oracle9*i*AS Web Cache Manager enables you to select one or more of the following:

`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
```

Notes:

- Oracle9*i*AS Web Cache does not interpret the values of these HTTP request headers. If the values for two pages are different, Oracle9*i*AS 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, Oracle9*i*AS Web Cache serves two pages.
- You can specify other HTTP request-header fields with the `Surrogate-Control` response-header field, as described in "Configuring Caching Attributes in Response Headers" on page 7-66.

See Also:

- "Multiple Versions of the Same Document" on page 2-12 for an overview
- "Configuring Rules for Multiple-Version Documents Containing HTTP Request Headers" on page 7-23 for additional configuration details

Session/Personalized Attribute Related Caching Rules	Select **None** to not have Oracle9*i*AS Web Cache cache or serve documents based on session or **personalized attribute** information contained within a cookie or embedded in a URL as a parameter. Select **Apply the following** to specify how Oracle9*i*AS Web Cache caches and serves documents with session or personalized attribute cookies or embedded URL parameters. If you do not see the session rules these documents require, then choose **General Configuration** > **Cacheability** > **Session/Personalized Caching Rules** to create a new rule. **See Also:** ■ "Controlling How Personalized Attribute Requests Are Served by the Cache" on page 2-20 or "Controlling How Session Requests Are Served by the Cache" on page 2-25 for an overview ■ "Configuring Session-Related or Personalized Attributed-Related Caching Rules" on page 7-38 for additional configuration details
Simple Personalization	Select **No** to not substitute session values used in **session-encoded URLs** or personalized attribute values enclosed within `<!-- WEBCACHETAG-->` and the `<!-- WEBCACHEEND-->` HTML tags Select **Yes** to substitute session or personalized attribute values. Oracle9*i*AS Web Cache will replace the value information based on the value of the cookie or the embedded URL parameter. Oracle9*i*AS Web Cache then serves these pages for Web browser requests that contain the cookie or the embedded URL parameter. **See Also:** ■ "Substituting Session Information in Session-Encoded URLs" on page 2-23 or "Personalized Attributes" on page 2-17 for an overview ■ "Configuring Session Definitions and Rules for Session-Encoded URLs" on page 7-26 or "Configuring Personalized Attribute Definitions and Rules for Personalized Attributes" on page 7-28 for additional configuration details

HTTP Error Caching Enter the HTTP error codes you want Oracle9*i*AS Web Cache to cache. If you enter multiple codes, use a comma to separate them. If there is a problem on the **origin server** 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 8-6

11. Repeat Steps 3 through 11 for each caching rule.

Tip: In addition to or as an alternative to creating caching rules with Oracle9*i*AS Web Cache Manager, application developers can choose to store the many of the caching attributes in the header of an HTTP response message. See "Configuring Caching Attributes in Response Headers" on page 7-66 for details.

Once the caching rules are configured, prioritize them.

To assign priority to rules:

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

   Higher priority rules are processed first.

2. Apply changes and restart Oracle9*i*AS Web Cache:

   a. In the Oracle9*i*AS Web Cache Manager main window, choose **Apply Changes**.

   b. In the navigator pane, select **Administration** > **Operations**.

      The Operations page appears.

   c. In the Operations page, choose **Restart** to restart Oracle9*i*AS Web Cache.

Caching Rules Example

Table 7–4 illustrates how an administrator might set up caching rules for a catalog built on Oracle *i*Store 11*i* technology, which enables e-merchants to design, build, and publish their stores on the World Wide Web.

Table 7–4 Oracle iStore Caching Rules Example

Priority	URL Expression	HTTP Method(s)	Cache/Don't Cache	Description
1	`/html/ibeCCkdHdr.*\.jsp.*`	Get, Get with query string, POST	Don't Cache	Instructs Oracle9*i*AS Web Cache to not cache billing details
2	`/html/ibeCCkpOrdReview\.jsp.*`	Get, Get with query string, POST	Don't Cache	Instructs Oracle9*i*AS Web Cache to not cache review order page
3	`/html/ibeCA.*\.jsp.*`	Get, Get with query string, POST	Don't Cache	Instructs Oracle9*i*AS Web Cache to not cache the account page
4	`/html/beCPmdPmtBook.\.jsp.*`	Get, Get with query string, POST	Don't Cache	Instructs Oracle9*i*AS Web Cache to not cache the payment book
5	`/html/ibeCXpd.*\.jsp.*`	Get, Get with query string, POST	Don't Cache	Instructs Oracle9*i*AS Web Cache to not cache order confirmation
6	`/html/ibeCSl.*\.jsp.*`	Get, Get with query string, POST	Don't Cache	Instructs Oracle9*i*AS Web Cache to not cache shopping lists
7	`/html/ibeCSc.*\.jsp.*`	Get, Get with query string, POST	Don't Cache	Instructs Oracle9*i*AS Web Cache to not cache shopping cart details
8	`/html/ibeCOtdOrdSumMain\.jsp.*`	Get, Get with query string, POST	Don't Cache	Instructs Oracle9*i*AS Web Cache to not cache the order status in account pages
9	`/html/ibeCCtpBuyRoute\.jsp.*`	Get, Get with query string	Cache	Instructs Oracle9*i*AS Web Cache to cache the buy routing page
10	`/html/ibeCZzpHome.*\.jsp$`	Get, Get with query string	Cache	Instructs Oracle9*i*AS Web Cache to cache the home page
11	`/html/ibeCCtpSctDspRte.jsp`	Get, Get with query string	Cache	Instructs Oracle9*i*AS Web Cache to cache the section routing page
12	`/html/ibeCCtpItmDspRte.jsp`	Get, Get with query string	Cache	Instructs Oracle9*i*AS Web Cache to cache the item routing page
13	`/html/ibeCSrdSrchResults.*\.jsp`	Get, Get with query string	Cache	Instructs Oracle9*i*AS Web Cache to cache the search results page
14	`/OA_MEDIA/.*`	Get, Get with query string	Cache	Instructs Oracle9*i*AS Web Cache to cache all images and multimedia
15	`/html/jtfucss.*\.css`	Get, Get with query string	Cache	Instructs Oracle9*i*AS Web Cache to cache the default style sheet

Note: Implementations of Oracle *i*Store can be customized. Therefore, these caching rules will not apply to all Oracle *i*Store deployments. See the Oracle *i*Store documentation for the most current information on implementing caching rules for Oracle *i*Store 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 Oracle9*i*AS Web Cache Manager.

   See Also: "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. In the navigator pane, select **General Configuration** > **Cacheability Rules** > **Expiration Rules**.

   The Expiration Rules page appears.

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

   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 `Expires` response-header field. This is the default. In order to utilize this option, documents must be programmed to use the HTTP `Expires` response-header field.

While the first two options enable you to set expiration for Oracle9*i*AS Web Cache-specific rules, the third option recognizes the expiration policy established for the documents already programmed with an HTTP `Expires` response-header field.

5. In the **After Expiration** section, specify how you want Oracle9*i*AS Web Cache to process documents once they have expired:

Remove immediately	Select this option to have Oracle9*i*AS 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 Oracle9*i*AS 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 to documents 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 caching rule, as described in "Configuring Caching Rules" on page 7-8. In Step 10 of the procedure, select an expiration rule in the **Expiration Rule** row of the Create Cacheability Rule dialog box.
10. Apply changes and restart Oracle9*i*AS Web Cache:
    a. In the Oracle9*i*AS Web Cache Manager main window, choose **Apply Changes**.
    b. In the navigator pane, select **Administration** > **Operations**.

       The Operations page appears.
    c. In the Operations page, choose **Restart** to restart Oracle9*i*AS Web Cache.

Configuring Rules for Multiple-Version Documents Containing Cookies

See Also: "Multiple Versions of the Same Document" on page 2-12 for an overview and an example scenario

You can specify which category cookies whose values Oracle9*i*AS Web Cache will use to cache and identify multiple-version documents.

To specify cookie values for multiple-version URLs:

1. Start Oracle9*i*AS Web Cache Manager.

   See Also: "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. In the navigator pane, select **General Configuration** > **Cacheability Rules** > **Multiple Documents with the Same Selector by Cookies**.

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

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 contain this cookie. This option enables Oracle9*i*AS Web Cache to serve documents from the cache for browser requests that do not 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 caching rule, as described in "Configuring Caching Rules" on page 7-8. 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 Oracle9*i*AS Web Cache:
    a. In the Oracle9*i*AS Web Cache Manager main window, choose **Apply Changes**.
    b. In the navigator pane, select **Administration** > **Operations**.

       The Operations page appears.
    c. In the Operations page, choose **Restart** to restart Oracle9*i*AS Web Cache.

Configuring Rules for Multiple-Version Documents Containing HTTP Request Headers

See Also: "Multiple Versions of the Same Document" on page 2-12 for an overview and an example scenario

You can specify which HTTP request headers whose values Oracle9*i*AS 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 Oracle9*i*AS 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 Caching Rules" on page 7-8

Configuring Session Definitions to Exclude the Value of URL Parameters

You can configure Oracle9*i*AS Web Cache to ignore the value of embedded URL parameters so that one version of a page can served to multiple users.

> **See Also:** "Ignoring the Value of Embedded URL Parameters" on page 2-22 for an overview and an example scenario

To ignore the value of URL parameters:

1. Start Oracle9*i*AS Web Cache Manager.

   > **See Also:** "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. Create session definitions for those pages that support embedded URL parameters:

   a. In the navigator pane, select **General Configuration** > **Session Management** > **Session/Personalized Attribute Definitions**.

      The Session / Personalized Attribute Definitions page appears.

   b. From the **For Site** list, select the Web site for which to create site-specific session definitions.

   c. Choose **Add** or **Create**.

      The Create Session / Personalized Attribute Definition dialog box appears.

   d. Select either **This Site Only** or **For All Sites.**

   e. In the **Session/Attribute** field, enter an easy-to-remember unique name for the session.

   f. Enter the embedded URL parameter in the **URL Parameter** field.

      Note: You can specify up to 20 definitions for each page.

      ---

   g. In the **Default Value (Optional)** field, enter a default string that Oracle9*i*AS Web Cache will use for the cookie or embedded URL parameter value.

      Oracle9*i*AS Web Cache uses the default string for those requests without the parameter information. For these requests, Oracle9*i*AS Web Cache substitutes the session information with the default string. The string

defaults to `default`. If you want to instead require that the request obtain the embedded URL parameter settings from the origin server, perform Step 3.

h. In the **Comment** field, enter a description for the definition.

i. Choose **Submit**.

3. If you want to require that the request obtain the embedded URL parameter settings from the origin server, perform these additional steps:

   a. Create a session-related caching rule for the page to track the session, as described in "Configuring Session-Related or Personalized Attributed-Related Caching Rules" on page 7-38.

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

   c. In Step 6b of the procedure, select **NO** as the response.

4. Apply changes and restart Oracle9*i*AS Web Cache:

   a. In the Oracle9*i*AS Web Cache Manager main window, choose **Apply Changes**.

   b. In the navigator pane, select **Administration** > **Operations**.

      The Operations page appears.

   c. In the Operations page, choose **Restart** to restart Oracle9*i*AS Web Cache.

Configuring Session Definitions and Rules for Session-Encoded URLs

You can specify caching rules for personalized pages that use session-encoded URLs. Session-encoded URLs enable Web sites to keep track of user sessions through session information contained within `<A HREF=...>` HTML tags. You can configure Oracle9*i*AS Web Cache to substitute session information for one user with another based on the session information contained within a cookie or an embedded URL parameter.

> **See Also:** "Substituting Session Information in Session-Encoded URLs" on page 2-23 for an overview and an example scenario

To cache instructions for substituting session information in session-encoded URLs.

1. Start Oracle9*i*AS Web Cache Manager.

   > **See Also:** "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. As necessary, create session definitions for those pages with session-encoded URLs:

   a. In the navigator pane, select **General Configuration** > **Session Management** > **Session/Personalized Attribute Definitions**.

      The Session / Personalized Attribute Definitions page appears.

   b. From the **For Site** list, select the Web site for which to create site-specific session definitions.

   c. Choose **Add** or **Create**.

      The Create Session / Personalized Attribute Definition dialog box appears.

   d. Select either **This Site Only** or **For All Sites.**

   e. In the **Session/Attribute** field, enter an easy-to-remember unique name for the session. You can use the same session definition used for ignoring a URL parameter.

   f. Enter the cookie name in the **Cookie Name** field and 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 session substitution. If they support different substitutions, create separate session definitions. You can specify up to 20 definitions for each page.

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

g. In the **Default Value (Optional)** field, enter a default string that Oracle9*i*AS Web Cache will use for the cookie or embedded URL parameter value.

   Oracle9*i*AS Web Cache uses the default string for those requests without the cookie or parameter information. For these requests, Oracle9*i*AS Web Cache substitutes the session information with the default string. The string defaults to `default`. If you want to instead require that the request get the cookie or embedded URL parameter settings from the origin server, perform Step 4.

h. In the **Comment** field, enter a description for the definition.

i. Choose **Submit**.

3. Create a caching rule for the documents that use the session, as described in "Configuring Caching Rules" on page 7-8.

   In Step 10 of the procedure, select **Yes** in the **Simple Personalization** row of the Edit / Create Cacheability Rule dialog box to substitute session information in session-encoded URLs.

4. If you want to require that the request get the cookie or embedded URL parameter settings from the origin server, perform these additional steps:

   a. Create a session-related caching rule for the page to track the session, as described in "Configuring Session-Related or Personalized Attributed-Related Caching Rules" on page 7-38.

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

   c. In Step 6b of the procedure, select **NO** as the response.

5. Apply changes and restart Oracle9*i*AS Web Cache:

   a. In the Oracle9*i*AS Web Cache Manager main window, choose **Apply Changes**.

   b. In the navigator pane, select **Administration** > **Operations**.

      The Operations page appears.

   c. In the Operations page, choose **Restart** to restart Oracle9*i*AS Web Cache.

Configuring Personalized Attribute Definitions and Rules for Personalized Attributes

You can specify caching rules for personalized pages that use personalized attributes. 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 Oracle9*i*AS Web Cache to substitute the value of personalized attributes contained within `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags based on the information contained within a cookie or an embedded URL parameter.

> **See Also:** "Personalized Attributes" on page 2-17 for an overview and an example scenario

To create rules for personalized pages:

1. Start Oracle9*i*AS Web Cache Manager.

   > **See Also:** "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. As necessary, create attribute definitions for those pages with personalized attributes:

   a. In the navigator pane, select **General Configuration** > **Session Management** > **Session/Personalized Attribute Definitions**.

      The Session / Personalized Attribute Definitions page appears.

   b. From the **For Site** list, select the Web site for which to create site-specific personalized attribute definitions.

   c. Choose **Add** or **Create**.

      The Create Session / Personalized Attribute Definition dialog box appears.

   d. Select either **This Site Only** or **For All Sites.**

   e. In the **Session/Attribute** field, enter an easy-to-remember unique name for the attribute.

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

   f. Enter the cookie name in the **Cookie Name** field and 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 substitution. If they support different substitutions, create separate personalized definitions. You can specify up to 20 definitions for each page.

Notes:

- Ensure that the size of cookies is not greater than 3 KB.
- You can also substitute session values between the `<!--WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags. To substitute session values, enter the session cookie in the **Cookie Name** field or the session parameter in the **URL Parameter** field.

---

g. In the **Default Value (Optional)** field, enter a default string that Oracle9*i*AS Web Cache will use for the cookie or embedded URL parameter value.

Oracle9*i*AS Web Cache uses the default string for those requests without the cookie or parameter information. For these requests, Oracle9*i*AS Web Cache substitutes the personalized attribute with the default string. The string defaults to `default`. If you want to instead require that the request get the cookie or embedded URL parameter settings from the origin server, perform Step 4.

h. In the **Comment** field, enter a description for the definition.

i. Choose **Submit**.

3. Create a caching rule for the personalized pages, as described in "Configuring Caching Rules" on page 7-8.

In Step 10 of the procedure, select **Yes** in the **Simple Personalization** row of the Edit / Create Cacheability Rule dialog box to substitute information in personalized attributes.

4. If you want to require that the request get the personalized attribute cookie or embedded URL parameter settings from the origin server, perform these additional steps:

a. Create a session-related caching rule for the page to track the session, as described in "Configuring Session-Related or Personalized Attributed-Related Caching Rules" on page 7-38.

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

c. In Step 6b of the procedure, select **NO** as the response.

5. Apply changes and restart Oracle9*i*AS Web Cache:

   a. In the Oracle9*i*AS Web Cache Manager main window, choose **Apply Changes**.

   b. In the navigator pane, select **Administration** > **Operations**.

      The Operations page appears.

   c. In the Operations page, choose **Restart** to restart Oracle9*i*AS Web Cache.

6. Configure the pages that use personalized attributes with the tags <!-- 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 7-45.

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-->:

```
htp.p('<FORM ACTION="test" METHOD="GET">');
htp.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>');
```

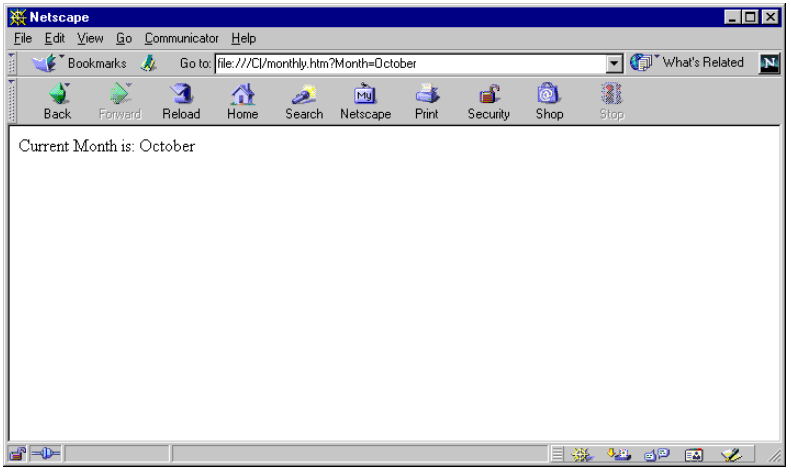
To achieve personalization within an HTML tag, use ESI.

See Also: "Example of Simple Personalization with Variable Expressions" on page 7-65

Example: Personalized Page Configuration

To understand how to cache personalized content, consider the HTML page `monthly.htm` in Figure 7–3.

Figure 7–3 monthly.htm



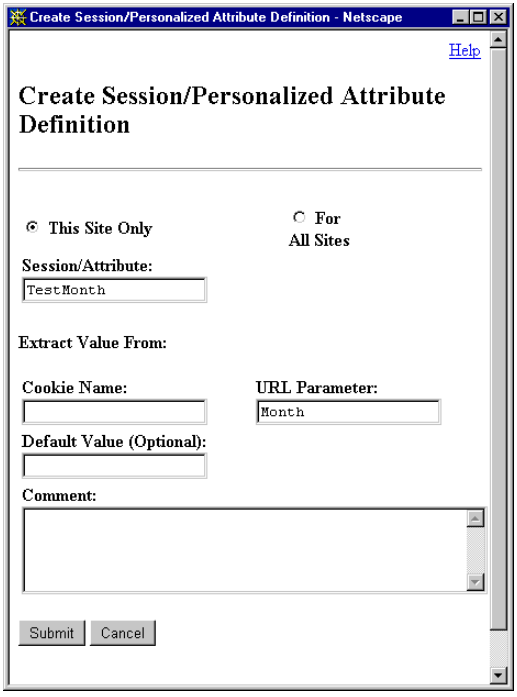
`October` is personalized content that can be substituted with other values.

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.

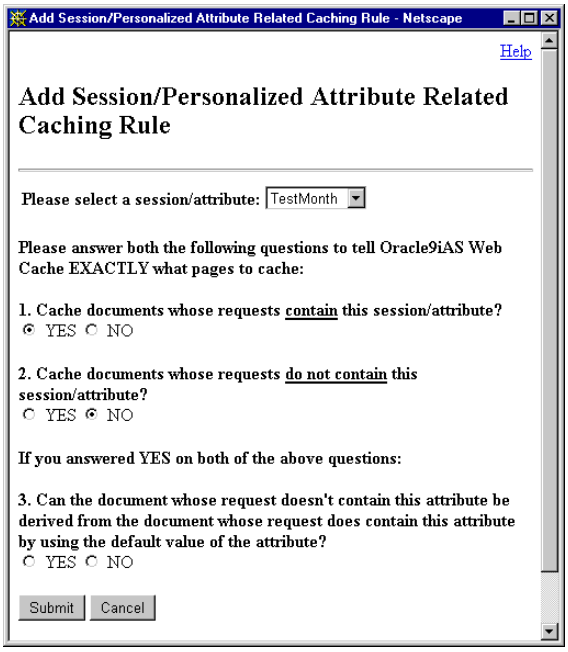
1. A personalized attribute of `TestMonth` was mapped to the embedded URL parameter `Month` in the Edit / Create Session / Personalized Attribute Definition dialog box.

Figure 7–4 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 7–5 Add Session/Personalized Attribute Related Caching Rule Dialog Box



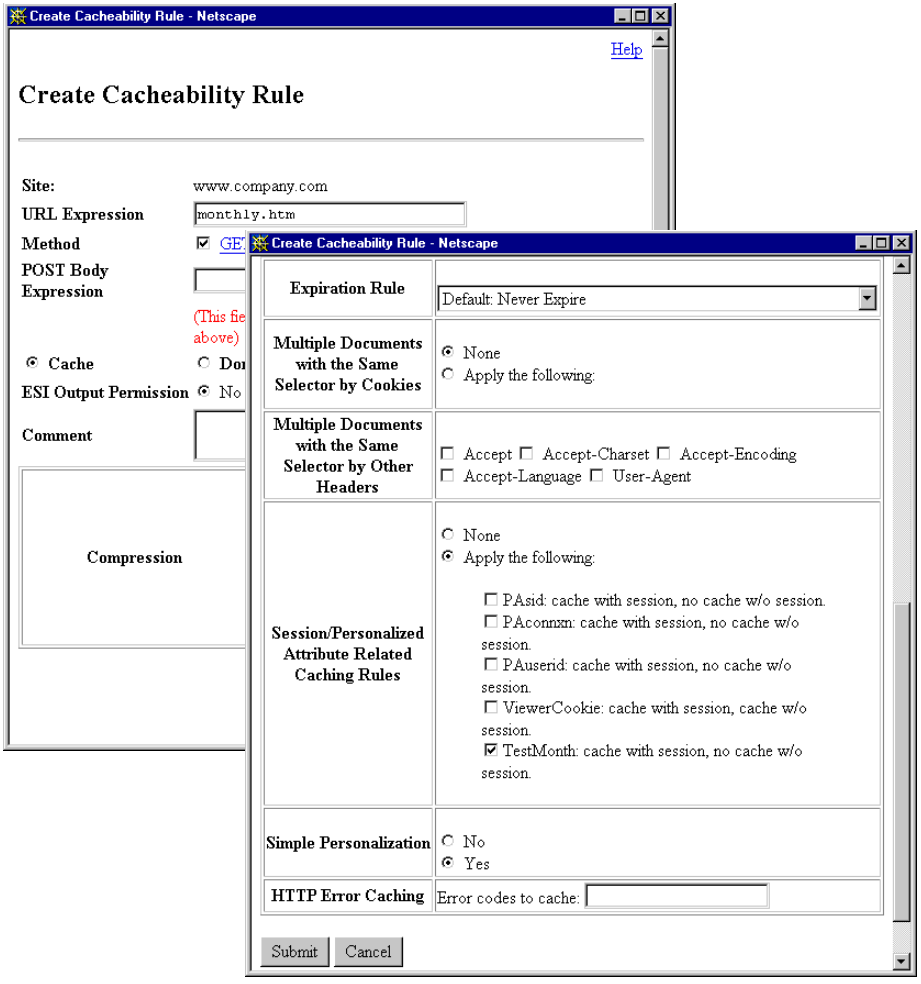
See Also: "Configuring Session-Related or Personalized Attributed-Related Caching Rules" on page 7-38 for more information about creating personalized attribute caching rules

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

```
Current Month is:
<!-- WEBCACHETAG="TestMonth"-->October<!-- WEBCACHEEND-->
```

4. A caching rule was 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** was chosen to cache substitution instructions for personalized attributes.

Figure 7–6 Create Cacheability Rule Dialog Box



5. The configuration changes are applied:

   a. In the Oracle9*i*AS Web Cache Manager main window, **Apply Changes** is chosen.

   b. In the Operations page, Oracle9*i*AS Web Cache is restarted.

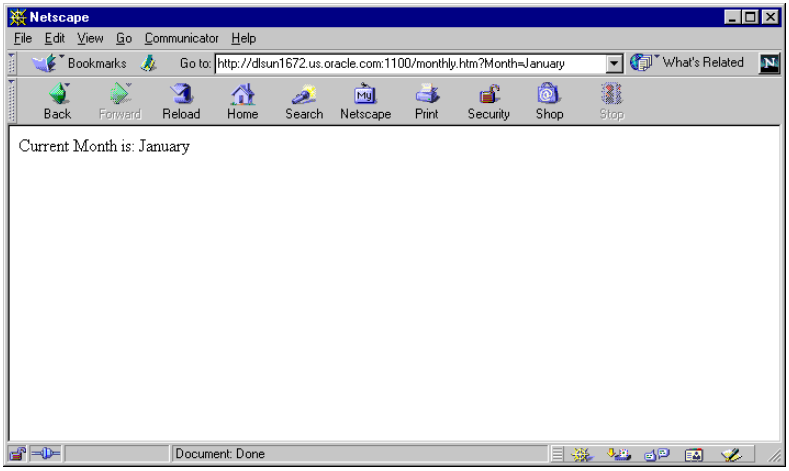
To verify that Oracle9*i*AS Web Cache was caching `monthly.htm`:

1. An initial request for `monthly.htm` at URL `monthly.htm?Month=October` was requested. Because the initial request was forwarded by Oracle9*i*AS 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`.

   Oracle9*i*AS Web Cache substituted `October` with the value of `January`.

Figure 7–7 monthly.htm When Cached



Configuring Session-Related or Personalized Attributed-Related Caching Rules

You can specify how Oracle9*i*AS Web Cache serves requests with the existence or nonexistence of session or personalized attribute cookies or embedded URL parameters.

See Also:

- "Controlling How Session Requests Are Served by the Cache" on page 2-25 for an overview of caching rules for sessions
- "Controlling How Personalized Attribute Requests Are Served by the Cache" on page 2-20 for an overview of caching rules for personalized attributes

To create caching rules for pages that support session cookies or personalized attributes:

1. Start Oracle9*i*AS Web Cache Manager.

   See Also: "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. In the navigator pane, select **General Configuration** > **Cacheability Rules** or **Session Management** > **Session/Personalized Attribute Related Caching Rules**.

   The Session / Personalized Attribute Related Caching Rules page appears.

3. In the Session / Personalized Attribute 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 or personalized attribute, and then proceed to Step 6.

Note: By default, Oracle9*i*AS Web Cache provides definitions of the following session identifiers that are commonly used by components of Oracle9*i* Application Server. The predefined site-specific 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`.
- `PAsid`, `PAconnxn`, `PAuserid`: `PAsid` is used for the Oracle9*i*AS Wireless session ID, `PAconnxn` is used for the Oracle9*i*AS Wireless connection ID, and `PAuserid` is used for the Oracle9*i*AS Wireless user ID. The embedded URL parameters are `PAsid`, `PAconnxn`, and `PAuserid`, respectively. No cookie names are used.

The predefined global session identifier is:

- `FoundationPersistentSessionID`: Used by Oracle9*i*AS Foundation Classes for persistent session tracking. The cookie name is `ESFSID`. There is no embedded URL parameter.

If the sessions or personalized attributes listed do not contain the definition you require, then choose **Cancel** to exit the Add Session / Personalized Attribute Related Caching Rules dialog box. Continue to Step 5.

5. Create a session or personalized attribute definition:
   a. In the navigator pane, select **General Configuration** > **Session Management** > **Session/Personalized Definitions**.

      The Session / Personalized Attribute Definitions page appears.

   b. From the **For Site** list, select the Web site for which to create site-specific definitions.

   c. Choose **Add** or **Create**.

      The Create Session / Personalized Attribute Definitions dialog box appears.

   d. Select either **This Site Only** or **For All Sites.**

   e. In the **Session/Attribute** field, enter an easy-to-remember unique name for the session or personalized attribute.

   f. Enter the cookie name in the **Cookie Name** field 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 or personalized attribute. If they support different sessions or personalized attributes, create separate definitions. You can specify up to 20 definitions for each page.

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

---

g. In the **Default Value (Optional)** field, enter a default string that Oracle9*i*AS Web Cache will use for the cookie or embedded URL parameter value.

Oracle9*i*AS Web Cache uses the default string for those requests without the cookie or parameter information. For these requests, Oracle9*i*AS Web Cache substitutes the session ID or personalized attribute information with the default string. The string defaults to `default`.

h. Optionally, in the **Comment** field, enter a description of the definition.

i. Choose **Submit**.

j. Repeats Steps 2 through 4.

6. Select **YES** or **NO** to the prompts:

Note: With an embedded URL parameter, Oracle9*i*AS Web Cache ignores the existence of an embedded URL parameter in the request. In other words, Oracle9*i*AS Web Cache will cache the response, even if the embedded parameter in the request does not match the embedded parameter in the response. To force the existence of the embedded URL parameter in the browser request, answer **YES** to the first prompt and **NO** to the second prompt.

---

a. For the prompt **1. Cache documents whose requests contain this session?** in the Add Session / Personalized Attribute Related Caching Rule dialog box, select either **YES** or **NO**:

– Select **YES** to cache versions of documents that use the session cookie or embedded URL parameter.

- Select **NO** to not cache versions of documents that use the session cookie or embedded URL parameter.

b. 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 cookie or embedded URL parameter. This selection enables Oracle9*i*AS Web Cache to serve documents from the cache for Web browser requests without the session or personalized attribute information.
- Select **NO** to not cache versions of documents that do not use the cookie or embedded URL parameter.

c. If you answered **YES** to the prompts described in Steps 6a and 6b, 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 cookie or embedded URL parameter, a default value is used.
- Select **NO** to cache two different versions of the document. Oracle9*i*AS Web Cache serves one version to those requests that support the cookie or the embedded parameter and serves the other version to those requests that do not support the cookie or embedded parameter.

7. In the Add Session / Personalized Attribute Related Caching Rule dialog box, choose **Submit**.

8. Associate the 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.

If the selector you require does not exist, then create a caching rule for the pages the support session or personalized attributed cookie or embedded URL parameter, as described in "Configuring Caching Rules" on page 7-8. 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.

9. Repeat Steps 3 through 8 for each rule.
10. Apply changes and restart Oracle9*i*AS Web Cache:
    a. In the Oracle9*i*AS Web Cache Manager main window, choose **Apply Changes**.
    b. In the navigator pane, select **Administration** > **Operations**.

       The Operations page appears.
    c. In the Operations page, choose **Restart** to restart Oracle9*i*AS Web Cache.

Configuring Caching Rules for Popular Pages with Session Establishment

Some Web sites require users to have sessions while surfing most pages. If you want to preserve the session requirement, then create a Session / Personalized Attribute Related Caching Rule for those pages. This way, a request without a session will always be served by the origin 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. Configure the origin server to create a session when the blank page is requested without a session cookie.
3. Create a caching rule for the real entry page and the blank page, as described in "Configuring Caching Rules" on page 7-8.

   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.

With this configuration, all initial user requests to the entry URL first go to the blank page, which requires minimal resources to generate. The browsers receive the response and session establishment from the application Web server. Subsequent redirected requests to the entry page will carry the session, enabling the entry page to be served out of the cache.

Another solution to this issue is to use a JavaScript that sets a session cookie for the pages requiring sessions:

1. Create a JavaScript that sets a session cookie when one does not exist.
2. Add the JavaScript to each of the pages that require the session.
3. Create caching rules for the JavaScript and the session pages, as described in "Configuring Caching Rules" on page 7-8.

Note: Using the JavaScript solution, it is not necessary to create a Session / Personalized Attribute Related Caching Rule for the pages requiring sessions.

Configuring Pages for Content Assembly and Partial Page Caching

See Also: "Content Assembly and Partial Page Caching" on page 2-26 for an overview of partial page caching

This section describes how to enable dynamic assembly of Web pages of 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. Configure the template page as follows:
   a. Use ESI markup tags in the template to fetch and include the fragments.

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 7-45.

---

b. In the template page, use a `Surrogate-Control` response-header field in the HTTP response message. For example:

```
Surrogate-Control: max-age=30+60, content="ORAESI/9.0.2"
```

c. If the `Surrogate-Control` response-header field does not include all the caching attributes required for the template page, then create a caching rule for the page.

2. Configure the fetchable fragments:

- If the fetchable fragments use ESI markup tags, then use a `Surrogate-Control` response-header field in the HTTP response message.
- If the `Surrogate-Control` response-header field does not include all the caching attributes required for the fragment, then create a caching rule for the fragment.

See Also:

- Appendix D, "Edge Side Includes Language" for further information about ESI markup tags
- "Configuring Caching Attributes in Response Headers" on page 7-66 for further information about configuring the `Surrogate-Control` response-header field
- "Configuring Caching Rules" on page 7-8 for further information about configuring caching rules

Using ESI for Simple Personalization

You can use variable expressions to achieve the same substitution as personalized attributes 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 Personalized Attribute Definitions and Rules for Personalized Attributes" on page 7-28.

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-5
- "ESI vars Tag" on page D-31

Examples of ESI Usage

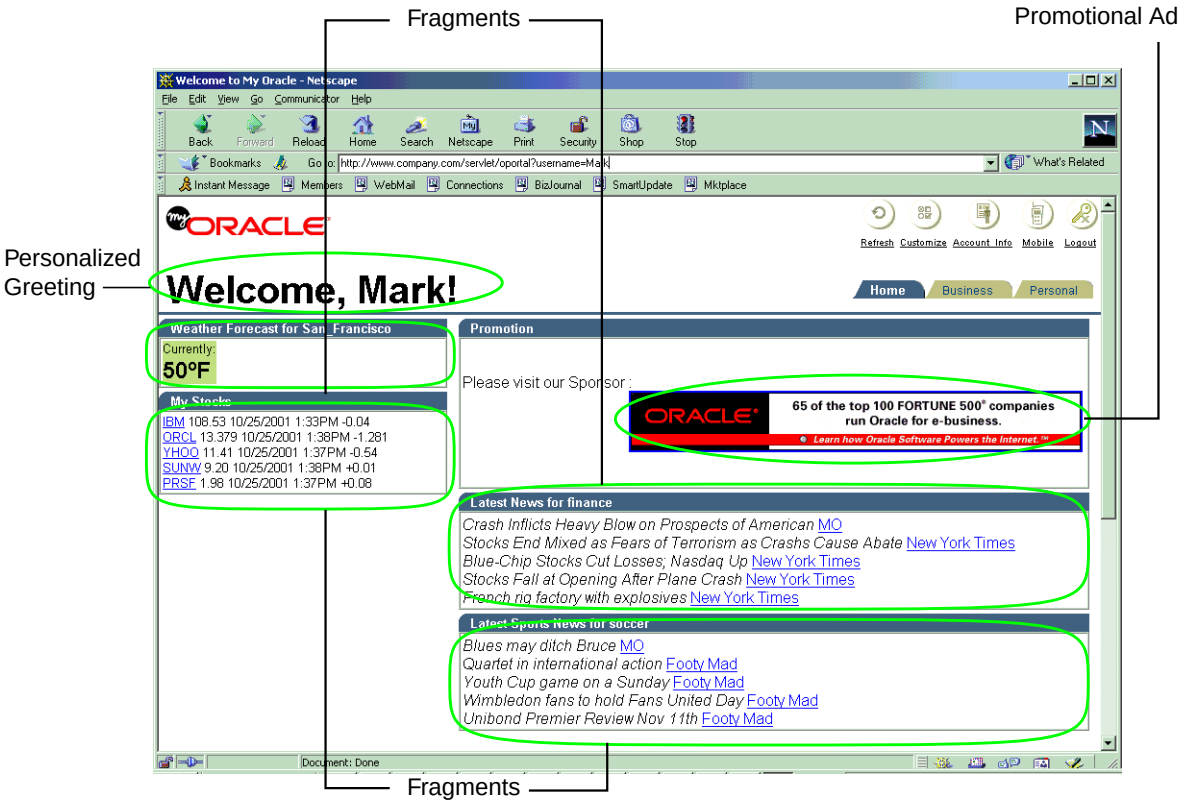
This section provides examples of ESI usage in the following topics:

- Example Portal Site Implementation
- Example of Simple Personalization with Variable Expressions

Example Portal Site Implementation

Figure 7–8 shows a portal site response page, `http://www.company.com/servlet/oportal?username=Mark`, for a registered user named Mark.

Figure 7–8 Portal Site Page



This page is assembled by Oracle9*i*AS Web Cache. A template page configured with ESI markup tags for a personalized greeting, weather, stocks, promotion advertisement, news, and sports fragments is assembled based on Mark's preferences. For example, since Mark chose San Francisco weather, the application looks up San Francisco weather information and puts it into the final full HTML page output. Because of its dynamic content, this page would not be cacheable. On

the other hand, with ESI markup tags, Oracle9*i*AS Web Cache assembles and caches most of the content.

The following sections describe how the template page and its fragments are implemented using `<esi:inline>` and `<esi:include>` tags:

- Portal Example Using inline Tags
- Portal Example Using Include Tags

Portal Example Using inline Tags

This section describes how `<esi:inline>` tag fragmentation and assembly can drastically increase the value of dynamic content caching for pages with that do not contain real-time elements. It shows how to apply the `<esi:inline>` tag for an existing application that supports non-fetchable fragments. The `<esi:inline>` tag helps reduce space consumption and improves cache hit ratios by isolating the dynamic content.

> **Note:** If an application supports independently fetchable fragments, it is possible to use the `<esi:inline>` for fetchable fragments by setting the `fetchable` attribute to `yes`. See "ESI inline Tag" on page D-5 for further information about the `fetchable` attribute.

To utilize the `<esi:inline>` tag, the logical fragments in `portal.esi` are marked with the `<esi:inline>` tags. The personalized greeting, Weather Forecast, My Stocks, Promotion campaign, Latest News, and Latest Sports News naturally become fragments because they have individual caching properties and can be shared. The My Stock fragment is further broken down into five sub-fragments, one for each stock quote. In addition, to achieve the maximum fragment sharing, the common HTML code sections between each two personalized fragments are also enclosed as ESI fragments and are given constant names, so that the varying template contains as little common data as possible.

Figure 7–9 shows `portal.esi` with `<esi:inline>` tags.

Figure 7–9 portal.esi with inline Tags

```
<esi:inline name="/Common_Fragment_1" >
<!-- First common fragment -->
<HTML>
...
<!-- Personalized Greeting With ESI variable -->
  Welcome, $(QUERY_STRING{username})!
</esi:inline>


<esi:inline name="/Weathers_San_Francisco" >
...
<!-- Personalized Weather Forecast -->
Weather Forecast for San Francisco
<TABLE>
  <TR>
    <TD>
      Currently: 63F
    </TD>
  </TR>
</TABLE>
</esi:inline>


<esi:inline name="/Common_Fragment_2" >
<!-- Second common fragment -->
...
</esi:inline>


<esi:inline name="/Stocks_$(QUERY_STRING{username})" >
<!-- Personalized Stock Quote Selections -->
<TABLE>
  <TR>
    <TD>
    <esi:inline name="/ticker_IBM">
      IBM 108.53 10/25/2001 1:33PM -0.04
    </esi:inline>
    <BR>
    <esi:inline name="/ticker_ORCL">
      ORCL 13.379 10/25/2001 1:38PM -1.281
    </esi:inline>
```

```
    <BR>
    <esi:inline name="/ticker_YHOO">
      YHOO 11.41 10/25/2001 1:37PM -0.54
    </esi:inline>
    <BR>
    <esi:inline name="/ticker_SUNW">
      SUNW 9.20 10/25/2001 1:38PM +0.01
    </esi:inline>
    <BR>
    <esi:inline name="/ticker_PRSF">
      PRSF 1.98 10/25/2001 1:37PM +0.08
    </esi:inline>
    <TD>
  </TR>
</TABLE>
</esi:inline>


<esi:inline name="/Common_Fragment_3">
<!-- Third common fragment -->
...
</esi:inline>


<esi:inline name="/ExternalAdvertisement">
<!-- External Advertisement -->
<TABLE>
  <TR>
    <TD>
    <a href="http://www.companyad.com/advert?promotionID=126532">
    <img src="http://www.companyad.com/advert_img?promotionID=126532">
    </a>
    </TD>
  </TR>
</TABLE>
</esi:inline>


<esi:inline name="/Common_Fragment_4">
<!-- Fourth common fragment -->
...
</esi:inline>
```

```
<esi:inline name="/Top_News_Finance">
<!-- Personalized Top News -->
Latest News for finance
<TABLE>
  <TR>
   Blue-Chip Stocks Cut Losses; Nasdaq Up MO
    Stocks Fall at Opening After Plane Crash New York Times
    French rig factory with explosives New York Times
    Volkswagen faces Brazil strike CNN Europe
    Airbuss reliability record BBC
  </TR>
</TABLE>
</esi:inline>


<esi:inline name="/Sports_News_Soccer" >
<!-- Personalized Sports News -->
Latest Sports News for Soccer

<TABLE>
  <TR>
  Hearts chief told to resign MO
  Owen and Gerrard fit for Blackburn game Ananova
  Hearts in positive AGM pledge Ananova
  Time for McIntosh to decide scotsman.com
  Bannon in call for calm as Gorgie faithfull gather to grill Robinson
  scotsman.com
  </TR>
</TABLE>
</esi:inline>


<esi:inline name="/Common_Fragment_5" >
...
</esi:inline>
```

Figure 7–10 shows the markup for the personalized greeting. The fragment is common to all personalized pages belonging to different users. Because the `<esi:inline>` tag assigns this fragment a constant name, a different user, such as John, would have the same fragment in his template with the same fragment name. Two fragments are shared if and only if their names are identical. This way, the same shared fragment in all templates only need a single update when it expires or is invalidated. `$(QUERY_STRING{username})` is an ESI environment variable that provide access to value of the `username`. This variable is used here because this application uses the `username` query string parameter to pass along the user's name. By using this variable, the first fragment becomes common to all users.

Figure 7–10 portal.esi Example with inline Tags: Personalized Greeting

```
<esi:inline name="/Common_Fragment_1" >
<!-- First common fragment -->
<HTML>
...
<!-- Personalized Greeting With ESI variable -->
  Welcome, $(QUERY_STRING{username})!
</esi:inline>
```

Figure 7–11 shows the markup for Weather Forecast. The fragment is unique to each city. Every template selecting the same city would share this fragment with Mark's page due to the fragment naming.

Figure 7–11 portal.esi Example with inline Tags: Weather Forecast

```
<esi:inline name="/Weathers_San_Francisco" >
<!-- Personalized Weather Forecast -->
Weather Forecast for San Francisco
<TABLE>
  <TR>
    <TD>
      Currently: 63F
    </TD>
  </TR>
</TABLE>
</esi:inline>
```

Figure 7–12 shows the markup for My Stocks. The stock quotes fragment encloses all stock picks in Mark's page. It is further divided into five sub-fragments, one for each stock pick, using nested <esi:inline> tags. Thus, Mark's ESI template references his stock selection fragment, which in turn references five particular stock pick fragments. While the stock picks are shared by many user's stock selection fragment, the stock selection fragment itself is also a template uniquely owned by Mark. This separates the unique information from shared information, maximizing the reduction of cache updates and space consumption of personal stock selection.

Figure 7–12 portal.esi Example: My Stocks Fragment

```
<esi:inline name="/Stocks_$(QUERY_STRING{username})" >
<!-- Personalized Stock Quote Selections -->
<TABLE>
  <TR>
    <TD>
    <esi:inline name="/ticker_IBM">
      IBM 108.53 10/25/2001 1:33PM -0.04
    </esi:inline>
    <BR>
    <esi:inline name="/ticker_ORCL">
      ORCL 13.379 10/25/2001 1:38PM -1.281
    </esi:inline>
    <BR>
    <esi:inline name="/ticker_YHOO">
      YHOO 11.41 10/25/2001 1:37PM -0.54
    </esi:inline>
    <BR>
    <esi:inline name="/ticker_SUNW">
      SUNW 9.20 10/25/2001 1:38PM +0.01
    </esi:inline>
    <BR>
    <esi:inline name="/ticker_PRSF">
      PRSF 1.98 10/25/2001 1:37PM +0.08
    </esi:inline>
    <TD>
  </TR>
</TABLE>
</esi:inline>
```

Figure 7–13 shows the markup for referencing an advertisement in the Promotion section. `promotionID` is the based on the user's identification.

Figure 7–13 portal.esi Example with inline Tags: Promotion

```
<esi:inline name="/ExternalAdvertisement">
<!-- External Advertisement -->
<TABLE>
  <TR>
    <TD>
    <a href="http://www.companyad.com/advert?promotionID=126532">
    <img src="http://www.companyad.com/advert_img?promotionID=126532">
    </a>
    </TD>
  </TR>
</TABLE>
</esi:inline>
```

Rotating advertisements that change in every response is an example of real- time content that renders little value in non-fetchable ESI `<esi:inline>` caching. Even the smallest portion of real-time content embedded as a non-fetchable ESI inline fragment would require the entire response to be regenerated and fetched, effectively creating cache misses all the time. To utilize ESI and dynamic content caching for these real-time fragments, use the `<esi:include>` tag.

> **See Also:** "Portal Example Using Include Tags" on page 7-55 for an example of using `<esi:include>` tag for real-time advertisements

The Latest News and Latest Sports News fragments are similar to the weather fragment. All the common areas are also defined as fragments. Although it is possible to leave them as part of the template, that would consume unnecessary storage space. Figure 7–14 shows the markup.

Figure 7–14 portal.esi Example with inline Tags: Latest News and Latest Sports News

```
<esi:inline name="/Top_News_Finance">
<!-- Personalized Top News -->
Latest News for finance
<TABLE>
  <TR>
   Blue-Chip Stocks Cut Losses; Nasdaq Up MO
    Stocks Fall at Opening After Plane Crash New York Times
    French rig factory with explosives New York Times
    Volkswagen faces Brazil strike CNN Europe
    Airbuss reliability record BBC
  </TR>
</TABLE>
</esi:inline>


<esi:inline name="/Sports_News_Soccer" >
<!-- Personalized Sports News -->
Latest Sports News for Soccer
<TABLE>
  <TR>
  Hearts chief told to resign MO
  Owen and Gerrard fit for Blackburn game Ananova
  Hearts in positive AGM pledge Ananova
  Time for McIntosh to decide scotsman.com
  Bannon in call for calm as Gorgie faithfull gather to grill Robinson
  scotsman.com
  </TR>
</TABLE>
</esi:inline>
```

Portal Example Using Include Tags

This section shows how the <esi:include> tag can be used for fragmentation and assembly of fetchable fragments whose content are not embedded in the template.

Figure 7–15 shows portal.esi with <esi:include> tags.

Figure 7–15 portal.esi with include Tags

```
<HTML>
...
<!-- Personal Profile -->
<esi:comment text="Profile refers to environment variables stored in
/servlet/GetProfile. GetProfile servlet enables access to a set of environment
variables with personal profile information."/>
<esi:environment src="/servlet/GetProfile?username=$(QUERY_STRING{username})"
name="Profile"/>
...


<!-- Personalized Greeting With ESI variable -->
<esi:vars>Welcome, $(QUERY_STRING{username})!</esi:vars>
...


<!-- Personalized Weather Forecast -->
<TABLE>
  <TR>
    <TD>
      <esi:include
src="/servlet/Weather?city=$(Profile{city})&state=$(Profile{state})"/>
    </TD>
  </TR>
</TABLE>
...
```

```
<!-- Personalized Stock Quote Selections -->
<TABLE>
  <TR>
    <TD>
      <esi:include src="/servlet/PersonalizedStockSelection?username=$(QUERY_
STRING{username})"/>
    </TD>
  </TR>
</TABLE>
...


<!-- External Advertisement -->
<TABLE>
  <TR>
    <TD>
      <esi:try>
        <esi:attempt>
          <esi:comment text="Include an ad"/>
          <esi:include src="/servlet/Advert"/>
       </esi:attempt>
       <esi:except>
         <esi:comment text="Just write an HTML link instead"/>
         <A HREF="http://www.oracle.com">http://www.oracle.com</a>
       </esi:except>
      </esi:try>
    </TD>
  </TR>
</TABLE>
...


<!-- Personalized Top News -->
Latest News for <esi:vars>$(Profile{news})</esi:vars>
<TABLE>
  <TR>
    <TD>
      <esi:choose>
        <esi:when test="$(Profile{news}) == 'internet'">
          <esi:include src="/servlet/News?type=Top&topic=internet"/>
        </esi:when>
        <esi:when test="$(Profile{news}) == 'finance'">
         <esi:include src="/servlet/News?type=Top&topic=business"/>
        </esi:when>
      <esi:otherwise>
```

```
      <esi:include src="/servlet/News?type=Top&topic=technology"/>
      </esi:otherwise>
      </esi:choose>
    </TD>
  </TR>
</TABLE>
...


<!-- Personalized Sports News -->
Latest Sports News for <esi:vars>$(Profile{sport})</esi:vars>
<TABLE>
  <TR>
    <TD>
      <esi:choose>
        <esi:when test="$(Profile{sport}) == 'golf'">
          <esi:include src="/servlet/News?type=Sports&topic=golf"/>
        </esi:when>
        <esi:when test="$(Profile{sport}) == 'soccer'">
          <esi:include src="/servlet/News?type=Sports&topic=soccer"/>
        </esi:when>
        <esi:when test="$(Profile{sport}) == 'basketball'">
          <esi:include src="/servlet/News?type=Sports&topic=basketball"/>
        </esi:when>
        <esi:when test="$(Profile{sport}) == 'baseball'">
          <esi:include src="/servlet/News?type=Sports&topic=baseball"/>
        </esi:when>
        <esi:otherwise>
          <esi:include src="/servlet/News?type=Sports&topic=soccer"/>
        </esi:otherwise>
      </esi:choose>
    </TD>
  </TR>
</TABLE>
```

Figure 7–16 specifies `Profile` to refer to the environment variables stored in `GetProfile`. `GetProfile` enables access to user profile variables, which are used as parameters in the included fragments:

Figure 7–16 portal.esi Example: Custom Profile Environment Variable Setting

```
<!-- Personal Profile -->
<esi:comment text="Profile refers to environment variables stored in
/servlet/GetProfile. GetProfile servlet enables access to a set of environment
variables with personal profile information."/>

<esi:environment src="/servlet/GetProfile?username=$(QUERY_STRING{username})"
name="Profile"/>
```

Figure 7–17 shows `GetProfile`, which provides access to the `city`, `state`, `news`, and `sports` environment variables.

Figure 7–17 portal.esi Example: GetProfile File with Environment Variables

```
<?xml version=1.0?>
<esi-environment esiversion="ORAESI/9.0.2">
  <city>San_Francisco</city>
  <state>CA</state>
  <news>finance</news>
  <sports>soccer</sports>
</esi-environment>
```

Figure 7–18 shows the markup for the personalized greeting `Welcome, Mark!`. 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 personalized greeting to be included in the cacheable template page.

Figure 7–18 portal.esi Example with vars tag: Personalized Greeting

```
<esi:vars>Welcome, $(QUERY_STRING{username})!</esi:vars>
```

Figure 7–19 shows the markup for Weather Forecast. Weather Forecast includes a servlet fragment name `Weather`, which uses the value of the user's `city` and `state` environment variables in `GetProfile` to display the correct weather forecast for the user. Because `GetProfile` has a value of `San Francisco` for the `city` environment variable and `California` for the `state` environment variable, the weather forecast is for San Francisco, California.

Figure 7–19 portal.esi Example with include Tags: Weather Forecast

```
<TABLE>
  <TR>
    <TD>
      <esi:include
src="/servlet/Weather?city=$(Profile{city})&state=$(Profile{state})"/>
    </TD>
  </TR>
</TABLE>
```

The markup for My Stocks is depicted in Figure 7–20. My Stocks includes a servlet fragment named `PersonalizedStockSelection`. The displayed stocks are based on the `userID` parameter encoded in the URL. `userID` is the registered user's unique ID.

Figure 7–20 portal.esi Example with include Tags: My Stocks Fragment

```
<TABLE>
  <TR>
    <TD>
      <esi:include src="/servlet/PersonalizedStockSelection?username=$(QUERY_
STRING{username})"/>
    </TD>
  </TR>
</TABLE>
```

The markup for the included fragment `PersonalizedStockSelection` is depicted in Figure 7–21. It includes fragments for five stock quotes: IBM, ORCL, YHOO, SUNW, and PRSF.

Figure 7–21 portal.esi Example: PersonalizedStockSelection Fragment for Mark

```
<TABLE>
  <TR>
    <TD>
    <BR>
    <esi:include src="Quote?symbol=IBM"/>
    <BR>
    <esi:include src="Quote?symbol=ORCL"/>
    <BR>
    <esi:include src="Quote?symbol=YHOO"/>
    <BR>
    <esi:include src="Quote?symbol=SUNW"/>
    <BR>
    <esi:include src="Quote?symbol=PRSF"/>
    <BR>
    </TD>
  </TR>
</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 7–22. It includes fragments for three stock quotes: IBM, ORCL, and SCO. As already described, IBM and ORCL are also shared by Mark. If Mark reloads the page first and caches the quotes, then the IBM and ORCL quotes for Scott are automatically refreshed.

Figure 7–22 portal.esi Example: PersonalizedStockSelection Fragment for Scott

```
<TABLE>
  <TR>
    <TD>
    <BR>
    <esi:include src="Quote?symbol=IBM"/>
    <BR>
    <esi:include src="Quote?symbol=ORCL"/>
    <BR>
    <esi:include src="Quote?symbol=SCO"/>
    <BR>
    </TD>
  </TR>
</TABLE>
```

Figure 7–23 shows the markup for rotating advertisements in the Promotion section. The advertisements rotates in the sense that the advertisement changes for each response. By separating the generation of the included image fragment response from the template page, Oracle9*i*AS Web Cache is able to cache the template and integrate the dynamic advertisement into the template.

Figure 7–23 portal.esi Example with include Tags: Promotion

```
<TABLE>
  <TR>
    <TD>
      <esi:try>
        <esi:attempt>
          <esi:comment text="Include an ad"/>
          <esi:include src="/servlet/Advert"/>
       </esi:attempt>
       <esi:except>
         <esi:comment text="Just write an HTML link instead"/>
         <A HREF="www.oracle.com">www.oracle.com</a>
        </esi:except>
      </esi:try>
    </TD>
  </TR>
</TABLE>
```

As shown in Figure 7–24, the response to the included image fragment for the banner is not cacheable. When a user requests this page, Oracle9*i*AS Web Cache sends the request to the application Web server to generate the banner. From the application Web server, `Advert` generates the banner for the request.

Figure 7–24 portal.esi Example: Rotating Banner Output

```
<TABLE>
  <TR>
    <TD>
      <A HREF="http://www.companyad.com/redirect?refID=11934502">
      <IMG src="http://www.companyad.com/advert_img?refID=11934502"></A>
    </TD>
  </TR>
</TABLE>
```

As shown in Figure 7–25, the next time the user reloads the page, `Advert` generates another banner for the request.

Figure 7–25 portal.esi Example: Rotating Banner Reload

```
<TABLE>
  <TR>
    <TD>
      <A HREF="http://www.companyad.com/redirect?refID=123456602">
      <IMG src="http://www.companyad.com/advert_img?refID=123456602"></A>
    </TD>
  </TR>
</TABLE>
```

The banner relies on alternate processing with the `<esi:try>` tag. If the servlet cannot run `Advert`, then a link to `www.oracle.com` appears in the banner's place.

Figure 7–26 shows the markup for Latest News and Latest Sports News:

- Latest News displays the news headlines based on the user's `news` category, `internet`, `finance`, or `technology`, by using conditional processing with the `<esi:choose>` tag. Because `GetProfile` has a value of `finance` for the `news` environment variable, the headlines displayed relate to finance, `/servlet/News?type=Top&topic=business`.
- Similarly, Latest Sports News displays the sports headlines based on the user's `sports` category, `golf`, `soccer`, `basketball`, `baseball`, or `soccer`, by using conditional processing. Because `GetProfile` has a value of `soccer` for the `sports` environment variable, the output includes headlines relating to soccer, `/servlet/News?type=Sports&topic=soccer`.

Figure 7–26 portal.esi Example with include Tags: Latest News and Sports Sections

```
Latest News for <esi:vars>$(Profile{news})</esi:vars>
<TABLE>
  <TR>
    <TD>
      <esi:choose>
        <esi:when test="$(Profile{news}) == 'internet'">
          <esi:include src="/servlet/News?type=Top&topic=internet"/>
        </esi:when>
        <esi:when test="$(Profile{news}) == 'finance'">
         <esi:include src="/servlet/News?type=Top&topic=business"/>
        </esi:when>
      <esi:otherwise>
      <esi:include src="/servlet/News?type=Top&topic=technology"/>
```

```
      </esi:otherwise>
      </esi:choose>
    </TD>
  </TR>
</TABLE>
...
<!-- Personalized Sports News -->
Latest Sports News for <esi:vars>$(Profile{sport})</esi:vars>

<TABLE>
  <TR>
    <TD>
      <esi:choose>
        <esi:when test="$(Profile{sport}) == 'golf'">
          <esi:include src="/servlet/News?type=Sports&topic=golf"/>
        </esi:when>
        <esi:when test="$(Profile{sport}) == 'soccer'">
          <esi:include src="/servlet/News?type=Sports&topic=soccer"/>
        </esi:when>
        <esi:when test="$(Profile{sport}) == 'basketball'">
          <esi:include src="/servlet/News?type=Sports&topic=basketball"/>
        </esi:when>
        <esi:when test="$(Profile{sport}) == 'baseball'">
          <esi:include src="/servlet/News?type=Sports&topic=baseball"/>
        </esi:when>
        <esi:otherwise>
          <esi:include src="/servlet/News?type=Sports&topic=soccer"/>
        </esi:otherwise>
      </esi:choose>
    </TD>
  </TR>
</TABLE>
```

Example of Simple Personalization with Variable Expressions

As described in Step 6 of "Configuring Personalized Attribute Definitions and Rules for Personalized Attributes" on page 7-28, 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 7–27. Its HTML code uses PL/SQL for an HTML form with a text box in it.

Figure 7–27 PL/SQL Code without Personalization

```
htp.p('<form action="test" method="GET">');
htp.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 7–28 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 7–28 PL/SQL Code with Personalization through ESI

```
htp.p('<form action="test" method="GET">');
htp.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 Caching Attributes in Response Headers

In addition to or as an alternative to creating caching rules with Oracle9*i*AS Web Cache Manager, application developers can choose to store the many of the caching attributes in the header of an HTTP response message. This feature enables the application Web server to override the settings configured through the Oracle9*i*AS Web Cache Manager interface, as well as allowing other third-party caches to use Oracle9*i*AS Web Cache caching attributes. All except the following attributes described in "Configuring Caching Rules" on page 7-8 are supported:

- ESI Output Permission
- Compression
- Session / Personalized Attributed Related Caching Rules
- HTTP Error Caching

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

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

Table 7–5 describes the supported control directives.

Table 7–5 Control Directives for Surrogate-Control

Control Directive	Description
`content`	Specify what kind of processing is required: ■ `"ORAESI/9.0.2"` to process ESI tags with Oracle proprietary additions for content assembly and partial page caching. `"ORAESI/9.0.2"` supports all the ESI tags provided by Oracle9*i*AS Web Cache in release 9.0.2 and release 9.0.3. ■ `"ESI/1.0"` to process standard ESI tags for content assembly and partial page caching ■ `"ESI-Inline/1.0"` to process `<esi:inline>` tags ■ `"webcache/1.0"` to process the `<!-- WEBCACHETAG-->` and `<!--WEBCACHEEND-->` tags for personalized attributes and session-encoded URLs `"ESI/1.0"` and `"ESI-Inline/1.0"` are subsets of `"ORAESI/9.0.2"`. In this release, you need to specify only `"ORAESI/9.0.2"` for ESI assembly or `"webcache/1.0"` for personalized attributes.
`max-age`	Specify to enable Oracle9*i*AS Web Cache to cache the document. Specify the time, in seconds, to expire the document after it enters the cache. Optionally, specify the time, in seconds, to remove the document from the cache after the expiration time. Use the following format: `max-age=`*`expiration_time`* `[+` *`removal_time`*`]` Usage notes: ■ The default removal time is 0 seconds ■ `max-age=infinity` specifies that the document never expires

Table 7–5 (Cont.) Control Directives for Surrogate-Control

Control Directive	Description
`no-store`	Specify Oracle9*i*AS Web Cache not to cache the document.
`no-store-remote`	Specify to now allow other ESI-compliant caches, such as Akamaii EdgeSuite, to cache the document. Specify to not allow other ESI-compliant caches, such as a third-party Content Delivery Network (CDN), to cache the document. The `no-store-remote` directive has similar semantics to the `no-store` 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 CDN. This directive is especially useful if you want to cache changing content locally, where invalidation propagation is immediate, but not in a distributed network of upstream ESI processors, where invalidation may take several minutes.
`vary`	Specify the HTTP request headers or category cookies from which Oracle9*i*AS Web Cache will use to cache and identify multiple-version documents. Use the following format: `vary=headers(`*`header header ...`*`); cookies(`*`cookie_name cookie_name ...`*`)` Usage notes: ■ `vary` accepts any valid HTTP request header ■ Use one or more spaces between the header and cookie names, and use zero or more spaces between the parenthesis and semicolons ■ Specify headers before cookies

Usage Notes

- Control directives are case sensitive
- `no-store` and `no-store-remote` are mutually exclusive
- `content="ORAESI/9.0.2"`, `content="ESI-Inline/1.0"`, `content="ESI/1.0"` are mutually exclusive with `content="webcache/1.0"`

Example Usage

In the following example, the `Surrogate-Control` response-header field specifies that the document is to expire 30 seconds after it enters the cache 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="ORAESI/9.0.2"
```

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

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

In the following example, the `Surrogate-Control` response-header field specifies ESI processing. It also specifies that the document is a multiple-version document that uses HTTP request headers of `Accept` and `MyCustomHeader` and cookies of `news` and `sports`.

```
Surrogate-Control: content="ORAESI/9.0.2", vary=headers(Accept
MyCustomHeader);cookies(news sports)
```

8

Administering Oracle9*i*AS Web Cache

This chapter explains how to perform administrative tasks to Oracle9*i*AS Web Cache.

This chapter contains these topics:

Starting and Stopping Oracle9*i*AS Web Cache

Anytime Oracle9*i*AS Web Cache's configuration is modified, you must stop and restart Oracle9*i*AS Web Cache. To start, stop, or restart Oracle9*i*AS Web Cache, use either Oracle9*i*AS Web Cache Manager or the `webcachectl` utility.

When you stop Oracle9*i*AS Web Cache, all objects are cleared from the cache. In addition, all statistics are cleared.

When you start Oracle9*i*AS Web Cache from the `webcachectl` utility, the **admin server process**, **cache server process**, and, if enabled, **auto-restart process** start.

- The `admin` server process manages the Oracle9*i*AS Web Cache Manager interface

  On Windows, the `admin` server process is represented by the `Oracle`*`HOME`*`_`*`NAME`*`WebCacheAdmin` service

- The `cache` server process to manage the cache

  On Windows, the `cache` server process is represented by the `Oracle`*`HOME`*`_`*`NAME`*`WebCache` service

- If enabled, the `auto-restart` process checks that the `cache` server process is running and automatically restarts the `cache` server process if it is not running

  Because the `auto-restart` process is dependent upon the `cache` server process, you administer it by starting, stopping, or restarting the `cache` server process.

  On Windows, the `auto-restart` process is represented by the `Oracle`*`HOME`*`_`*`NAME`*`WebCacheMon` service

  > **See Also:** "Task 6: Configure Auto-Restart Process Settings" on page 6-12 for instructions on enabling the `Oracle`*`HOME`*`_`*`NAME`*`WebCacheMon` service

When you start Oracle9*i*AS Web Cache from the Oracle9*i*AS Web Cache Manager, only the `cache` server process and, if enabled, the `auto-restart` process start. To initialize Oracle9*i*AS Web Cache for the first time, use the `webcachectl` utility to start all the processes.

To start, stop, or restart the processes:

Use Oracle9*i*AS Web Cache Manager...	Use the webcachectl Utility...

To start, stop, or restart the `cache` server and `auto-restart` processes:
1. Start Oracle9*i*AS Web Cache Manager.
See Also: "Starting Oracle9iAS Web Cache Manager" on page 5-3
2. In the navigator pane, select **Administration** > **Operations**.
The Operations page appears in the right pane.
3. In the Operations page, select the cache and choose **Start**, **Stop**, or **Restart**.
To perform the operation on one cache in a **cache cluster**:
Select one cache, choose **Selected Cache** from the **Operate On** field and then choose **Start**, **Stop**, or **Restart**.
To perform the operation on all caches in a cache cluster:
Choose **All Caches** from the **Operate On** field and then choose **Start**, **Stop**, or **Restart**.
	To start the `admin` server, `cache` server, and `auto-restart` processes:
	1. Determine the status of Oracle9*i*AS Web Cache. From the command line, enter:
	`webcachectl status`
	If the following message appears, Oracle9*i*AS Web Cache is not running. Continue to Step 2.
	`Oracle Web Cache admin server is NOT running.`
	`Oracle Web Cache auto-restart is NOT running.`
	`Oracle Web Cache cache server is NOT running.`
	If the following message appears, Oracle9*i*AS Web Cache is already running.
	`Oracle Web Cache admin server is running (pid=`*`pid`*`).`
	`Oracle Web Cache auto-restart monitor is running (pid=`*`pid`*`).`
	`Oracle Web Cache cache server is running (pid=`*`pid`*`).`
	2. Start the processes. From the command line, enter:
	`webcachectl start`
	To stop the `admin` server, `cache` server, and `auto-restart` processes, from the command line, enter:
	`webcachectl stop`
	To restart the `admin` server, `cache` server, and `auto-restart` processes, from the command line, enter:
	`webcachectl restart`
	See Also: "webcachectl Utility for Process Administration" on page 5-9 for a complete list of the `webcachectl` commands

On Windows, you can start or stop Oracle9*i*AS Web Cache through the Control Panel:

1. Select the **Services** icon in the Control Panel window.

   The Services window appears.

2. Select the `Oracle`*`HOME_NAME`*`WebCacheAdmin` service to start the `admin` server process, and then choose **Start** or **Stop**.

3. Select the `Oracle`*`HOME_NAME`*`WebCache` service to start the `cache` server process, and then choose **Start** or **Stop**.

   The `Oracle`*`HOME_NAME`*`WebCache` service also starts or stops the `Oracle`*`HOME_NAME`*`WebCacheMon` service, if it is enabled. You cannot select the `Oracle`*`HOME_NAME`*`WebCacheMon` service and start or stop it. It is dependent on the `Oracle`*`HOME_NAME`*`WebCache` service.

4. In the Services window, choose **Close.**

Propagating Configuration Changes to Cache Cluster Members

If you have made changes to the configuration of a cache cluster or if a **cache cluster member** is unreachable when Oracle9*i*AS Web Cache tries to propagate the configuration to it, you must propagate the configuration to that cluster member when it is reachable again. Then, you must restart the cache.

Oracle9*i*AS Web Cache keeps track of the configuration of all cluster members to ensure that all cluster members are using the same version of the configuration. It compares the configuration of the current cache (the cache to which you are connected) to that of the other cluster members.

To check that all cluster members are using the same configuration and to propagate the configuration, if necessary, perform following steps:

1. In the navigation pane, select **Administration** > **Operations.**

   The Operations page appears.

2. In the **Operation Needed** column of the table, check whether or not **Propagate Configuration** is noted for any cluster member.

   Propagate Configuration means that the configuration of the cluster member is different than the configuration of the current cache. You should verify that the configuration of the current cache is the most valid configuration before proceeding with propagation.

   If it is not, connect to the cache with the valid configuration and view the **Operations** page.

3. For each cluster member that needs the configuration propagated, select the cache. Then, in the **Operate On** field, choose **Selected cache** and choose **Propagate**. (Alternatively, to operate on all caches in the cluster, in the **Operate On** field, choose **All caches** and specify an interval to stagger the times of the operations, and then choose **Propagate**.)

   Oracle9*i*AS Web Cache propagates the configuration from the current cache to the selected cluster member. When the operation completes, the **Operation Needed** column in the Operations page indicates that the cache needs to be restarted.

4. To restart one cluster member:

   a. Select the cache.

   b. In the **Operate On** field, choose **Selected cache** and choose **Restart.** (Alternatively, to operate on all caches in the cluster, in the **Operate On** field, choose **All caches** and specify an interval to stagger the times of the operations, and then choose **Restart**.)

Invalidating Documents in the Cache

To invalidate documents in the cache, send an HTTP `POST` message from the `invalidator` user through an invalidation listening port.

The `invalidator` user is an administrator authorized to send invalidation messages. In a **cache hierarchy** of Oracle9*i*AS Web Cache servers, the **central cache** or **provider cache** uses the `invalidator` user name and password of the remote or subscriber Oracle9*i*AS Web Cache server. The invalidation request specifies the documents to invalidate, as well as the site host name of the documents.

Note: In order for automatic propagation of invalidation messages to work, Oracle9*i*AS Web Cache passes the encoded `invalidator` password in the page request between the central and **remote cache** or the provider and **subscriber cache**. This HTTP traffic is susceptible to network sniffing. If the network is unprotected and insecure, configure HTTPS ports as follows:

1. In the Listening Ports page (**Cache-Specific Configuration** > **Listening Ports**) of the central or provider cache, disable the default HTTP port. An HTTPS port is already configured by default. See "Task 3: Configure Oracle9iAS Web Cache with Listening Ports for Incoming Browser Requests" on page 6-6.
2. In the Operations page (**Cache-Specific Configuration** > **Operations Ports**) of the remote or subscriber cache, disable the default HTTP port and configure an HTTPS port in its place. See "Task 5: Configure Oracle9iAS Web Cache with Operations Ports" on page 6-9.

This section contains the following invalidation-related topics:

- Sending Invalidation Requests
- Invalidation Examples

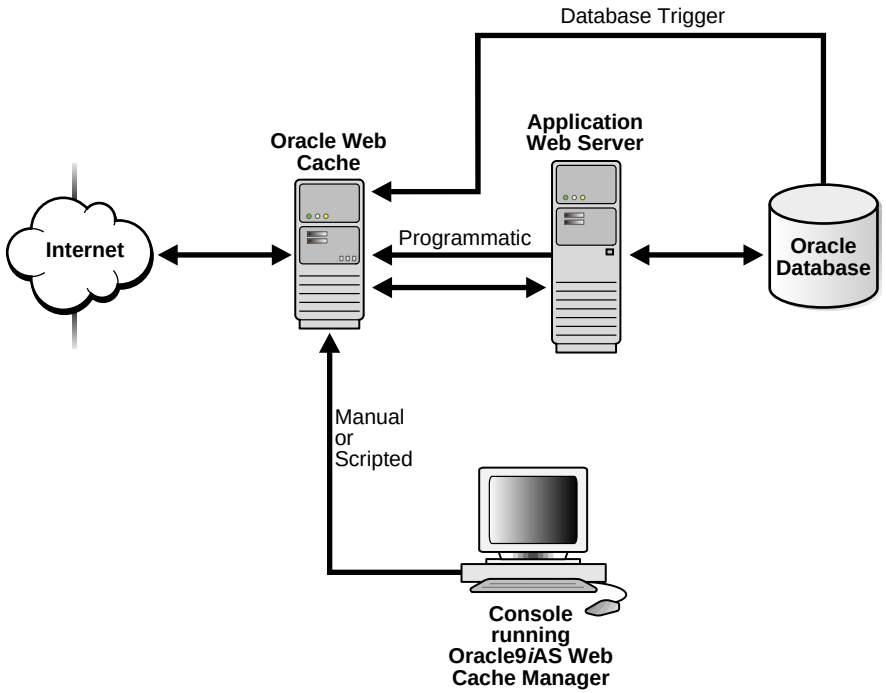
See Also: "Invalidation in Hierarchies" on page 2-7

Sending Invalidation Requests

Invalidation requests are HTTP POST requests written in **Extensible Markup Language (XML)** syntax. The contents of the XML request body instructs the cache which URLs to mark as invalid. As shown in Figure 8–1, invalidation requests can be sent using one of the following methods:

- Manually, using either Oracle9*i*AS Web Cache Manager or `telnet`
- Automatically, using database triggers, scripts, or application logic

Figure 8–1 Invalidation



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

- Manual Invalidation Using Telnet
- Manual Invalidation Using Oracle9iAS Web Cache Manager
- Automatic Invalidation Using Applications
- Automatic Invalidation Using Database Triggers
- Automatic Invalidation Using Scripts
- Automatic Invalidation Using Scripts

Manual Invalidation Using Telnet

When you send an invalidation request with an HTTP `POST` request, you specify the host name of Oracle9*i*AS Web Cache, the invalidation listening port number, and the invalidation request.

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

1. Connect to Oracle9*i*AS Web Cache at the invalidation listening port:

   ```
   telnet web_cache_host invalidation_port
   ```

2. Specify a `POST` message header and authenticate the `invalidator` user 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
- `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 `EncodeBase64.java` script to generate the Base64 string for `invalidator:`*`invalidator_password`*
- "Task 2: Modify Security Settings" on page 5-3 for further information about changing the invalidation password

3. Enter one carriage return.
4. Send the invalidation request with XML syntax.

Invalidation request syntax is described in the following sections:

- Invalidation Request Syntax
- Invalidation Preview Request Syntax
- Invalidation Response Syntax
- Invalidation Preview Response Syntax

Invalidation Request Syntax

Use the following syntax to invalidate documents 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-1.1">
   <SYSTEM>
    <SYSTEMINFO NAME="name" VALUE="value"/>
   </SYSTEM>
   <OBJECT>
     <BASICSELECTOR URI="URL"/>
     <ACTION REMOVALTTL="TTL"/>
     <INFO VALUE="value"/>
   </OBJECT>
</INVALIDATION>
```

Use the following syntax to invalidate documents based on more advanced invalidation selectors:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.1">
   <SYSTEM>
    <SYSTEMINFO NAME="name" VALUE="value"/>
   </SYSTEM>
   <OBJECT>
     <ADVANCEDSELECTOR URIPREFIX="prefix"
                       URIEXP="URL_expression"
                       HOST="host_name:port"
                       METHOD="HTTP_request_method"
                       BODYEXP="HTTP_body"/>
      <COOKIE NAME="cookie_name" VALUE="value"/>
      <HEADER NAME="HTTP_request_header" VALUE="value"/>
      <OTHER NAME="URI|BODY|QUERYSTRING_PARAMETER" TYPE="SUBSTRING|REGEX"
       VALUE="value"/>
     </ADVANCEDSELECTOR>
     <ACTION REMOVALTTL="TTL"/>
     <INFO VALUE="value"/>
   </OBJECT>
</INVALIDATION>
```

The body of a valid invalidation request must begin with the following:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
```

The first line denotes version 1.0 of XML. The second line denotes the request is an invalidation request using the `WCSinvalidation.dtd` file as the XML document type. `WCSinvalidation.dtd` is the Document Type Definition (DTD) that defines the grammar of invalidation requests and responses.

Notes:

- No white space is allowed before "`<?xml`".
- If an application is sharing invalidation requests with a third-party XML parser, replace `"internal:///WCSinvalidation.dtd"` with the following path:

  `"http://www.oracle.com/webcache/90200/WCSinvalidation.dtd"`

The root element INVALIDATION contains one or more of the attributes and elements described in Table 8–1.

Table 8–1 INVALIDATION Elements and Attributes

Invalidation Element/Attribute	Description
VERSION attribute	Required attribute in the INVALIDATION element Denote the version of the WCSinvalidation.dtd file to use as the XML document type. For versions 9.0.2 and 9.0.3, always use VERSION="WCS-1.1", unless you require existing applications to remain unchanged. For these applications, you can use VERSION="WCS-1.0", but the new invalidation functionality will not be available.
SYSTEM element	Optional element in the INVALIDATION element The SYSTEM element is optional and intended only for cache cluster configurations. The SYSTEM element requires the SYSTEMINFO element. SYSTEMINFO element The possible NAME/VALUE pair in a request is as follows: NAME="WCS_PROPAGATE" VALUE="TRUE|FALSE" This pair specifies whether or not invalidation requests are propagated to cache cluster members. If WCS_PROPAGATE is TRUE, it overrides the setting for invalidation propagation in the configuration. If WCS_PROPAGATE is FALSE, it uses the setting specified in the configuration. The default is FALSE.

Table 8–1 (Cont.) INVALIDATION Elements and Attributes

Invalidation Element/Attribute	Description
OBJECT element	Required element in the invalidation request. You can specify more than one OBJECT element in the request.
BASICSELECTOR element	URI attribute Required attribute of the BASICSELECTOR element. Specify the URL of the documents to be invalidated. Use one of the following formats: http://*host_name*:*port*/*path*/*filename* https://*host_name*:*port*/*path*/*filename* *host_name*:*port* is not required if the administrator user is sending the request.
ADVANCEDSELECTOR element	URIPREFIX attribute Required attribute of the ADVANCEDSELECTOR element Specify the prefix path of the documents to be invalidated. The prefix path must begin with http|https://*host_name*:*port*/*path*/*filename* or "/" and end with "/". *host_name*:*port* is required if the HOST attribute is not specified and the invalidator user is sending the request. 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 (\). URIEXP attribute Optional attribute of the ADVANCEDSELECTOR element Specify the URL of the documents to be invalidated underneath the URIPREFIX. If no value is entered, then everything under the URIPREFIX will be matched. Regular expression characters are permitted. To interpret these characters literally, escape them with a backslash (\). **Notes:** The request URL that browsers send to Oracle9*i*AS Web Cache and the URL that Oracle9*i*AS Web Cache uses internally for that request are different. When Oracle9*i*AS Web Cache serves a page request, it alphabetically sorts any embedded URL parameters of the URL. However, the invalidation requests are matched against only the internal representation of the URL in which any embedded URL parameters are sorted. To ensure invalidation requests are matched correctly, sort and enter the embedded URL parameters alphabetically. When the invalidation request is sent, Oracle9*i*AS Web Cache performs a regular expression match of URIEXP. This can take processing time. As an alternative, you can use the OTHER element to specify a substring match rather than a regular expression match.

Table 8–1 (Cont.) INVALIDATION Elements and Attributes

Invalidation Element/Attribute	Description
	HOST attribute This attribute is required if the URIPREFIX value does not include *host_name*:*port* and the invalidator user is sending the request. Specify the host name and port number of the site (*host_name*:*port*). Port 80 is the default port for HTTP and port 443 is the default port for HTTPS. METHOD attribute Optional attribute of the ADVANCEDSELECTOR element Specify **HTTP request method** (GET or POST) of the documents to be invalidated. BODYEXP attribute Optional attribute of the ADVANCEDSELECTOR element If the METHOD is POST, specify HTTP POST body of the documents to be invalidated. **Note:** When the invalidation request is sent, Oracle9*i*AS Web Cache performs a regular expression match of BODYEXP. This can take processing time. As an alternative, you can use the OTHER element to specify a substring match rather than a regular expression match. COOKIE element Optional element in the invalidation request NAME attribute Required attribute for the COOKIE element attribute 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. VALUE attribute Optional attribute for the COOKIE element Specify the value of the cookie. If no value is present, then only documents with the named cookie but without the value are invalidated.

Table 8–1 (Cont.) INVALIDATION Elements and Attributes

Invalidation Element/Attribute	Description
	HEADER element Optional element in the invalidation request NAME attribute Required attribute for the HEADER 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. VALUE attribute Optional attribute for the HEADER element. Specify the value of the header. OTHER element Optional element in the invalidation request NAME attribute: Required attribute of the OTHER element. NAME can have one of the following values: - URI to specify a match of the URL specified in VALUE - BODY to specify a match of the HTTP POST body - QUERYSTRING_PARAMETER to specify a match of an embedded URL parameter TYPE attribute: Required attribute of the OTHER element. TYPE can have one of the following values: - SUBSTRING to specify a substring match - REGEX to specify a regular expression match VALUE: Required attribute for the OTHER element. Specify the value of URI, BODY, or QUERYSTRING_PARAMETER. **See Also**: "Reducing Invalidation Overhead" on page 8-44 to optimize advanced invalidations

Table 8–1 (Cont.) INVALIDATION Elements and Attributes

Invalidation Element/Attribute	Description
ACTION element	Required element in the invalidation request REMOVALTTL attribute Optional attribute of the ACTION element Specify the maximum time that documents can reside in the cache before they are invalidated. The default is 0 seconds.
INFO element	Optional element in the invalidation request VALUE attribute Required attribute of the INFO element Specify a comment for the documents to be included in the invalidation result. After the invalidation request is complete, the message that contains the comment, along with the result of the invalidation, is written to the event log: <Invalidation>Invalidation with info '*INFO_comment*' has returned with status '*status*'; number of documents invalidated: '*number*'

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

Note: Oracle9*i*AS Web Cache continues to support invalidation requests 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>
```

See Also: "Invalidation Request and Response DTD" on page C-3 for further information about invalidation request syntax

Invalidation Preview Request Syntax

To test invalidation, use the following syntax to preview the list of BASICSELECTOR documents to be invalidated:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONPREVIEW SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONPREVIEW VERSION="WCS-1.1" STARTNUM="start_number" MAXNUM="max_
number">
     <BASICSELECTOR URI="URL"/>
</INVALIDATION>
```

Use the following syntax to preview the list of ADVANCEDSELECTOR documents to be invalidated:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONPREVIEW SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONPREVIEW VERSION="WCS-1.1" STARTNUM="start_number" MAXNUM="max_
number">
    <ADVANCEDSELECTOR URIPREFIX="prefix"
                      URIEXP="URL_expression"
                      HOST="host_name:port"
                      METHOD="HTTP_request_method"
                      BODYEXP="HTTP_body"
      <COOKIE NAME="cookie_name" VALUE="value"/>
      <HEADER NAME="HTTP_request_header" VALUE="value"/>
      <OTHER NAME="URI|BODY|QUERYSTRING_PARAMETER" TYPE="SUBSTRING|REGEX"
       VALUE="value"/>
    </ADVANCEDSELECTOR>
</INVALIDATION>
```

The body of a valid invalidation preview request must begin with the following:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONPREVIEW SYSTEM "internal:///WCSinvalidation.dtd">
```

The first line denotes version 1.0 of XML. The second line denotes the request is an invalidation preview request using the WCSinvalidation.dtd file as the XML document type.

The root element INVALIDATIONPREVIEW contains one or more of the attributes described in Table 8–2. BASICSELECTOR and ADVANCEDSELECTOR are described in Table 8–1 on page 8-12.

Table 8–2 INVALIDATIONPREVIEW Attributes

Invalidation Element/Attribute	Description
VERSION attribute	Required attribute in the INVALIDATIONPREVIEW element Denote VERSION="WCS-1.1" as the version of the WCSinvalidation.dtd file to use as the XML document type.
STARTNUM attribute	Required attribute in the INVALIDATIONPREVIEW element Enter the number representing the first document to be listed. Oracle9*i*AS Web Cache begins the count of documents with the number 0.
MAXNUM attribute	Required attribute in the INVALIDATIONPREVIEW element Enter the number of documents to be listed. If fewer documents than the number specified meet the invalidation criteria, Oracle9*i*AS Web Cache lists only the URLs for those documents that meet the criteria. If more documents than the number specified meet the invalidation criteria, Oracle9*i*AS Web Cache lists the URLs for the number of documents requested. It also returns the total number of documents that meet the invalidation criteria.

Invalidation Response Syntax

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

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

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

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONRESULT VERSION="WCS-1.0">
   <SYSTEM>
     <SYSTEMINFO NAME="name" VALUE="value"/>
   </SYSTEM>
   <OBJECTRESULT>
    <ADVANCEDSELECTOR URIPREFIX="prefix"
                      URIEXP="URL_expression"
                      HOST="host_name:port"
                      METHOD="HTTP_request_method"
                      BODYEXP="HTTP_body"/>
      <COOKIE NAME="cookie_name" VALUE="value"/>
      <HEADER NAME="HTTP_request_header" VALUE="value"/>
      <OTHER NAME="URI|BODY|QUERYSTRING_PARAMETER" TYPE="SUBSTRING|REGEX"
       VALUE="value"/>
     </ADVANCEDSELECTOR>
     <RESULT ID="ID" STATUS="status" NUMINV="number"/>
     <INFO VALUE="value"/>
   </OBJECTRESULT>
</INVALIDATIONRESULT>
```

The body of a valid invalidation response begins with the following:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT SYSTEM "internal:///WCSinvalidation.dtd">
```

The first line denotes version 1.0 of XML. The second line denotes the response is an invalidation response using the `WCSinvalidation.dtd` file as the XML document type.

The root element `INVALIDATIONRESULT` contains one or more of the attributes and elements described in Table 8–3. `BASICSELECTOR` and `ADVANCEDSELECTOR` are described in Table 8–1 on page 8-12.

Table 8–3 INVALIDATIONRESULT Elements and Attributes

Invalidation Element/Attribute	Description
`VERSION` attribute	Version number of the `WCSinvalidation.dtd` file to use as the XML document type
`SYSTEM` element	Optional element in the `INVALIDATIONRESULT` element. The `SYSTEM` element is optional and intended only for cache cluster configurations. The `SYSTEM` element requires the `SYSTEMINFO` element. ■ `SYSTEMINFO` element The possible `NAME`/`VALUE` pair in a request is as follows: `NAME="WCS_CACHE_NAME" VALUE="`*`string`*`"` This pair specifies the name of the cache.

Table 8–3 (Cont.) INVALIDATIONRESULT Elements and Attributes

Invalidation Element/Attribute	Description
RESULT element	ID attribute Sequence number of all the invalidation objects sent in the invalidation response. If there are multiple selectors specified in the invalidation request, then the sequence number starts at 1 for the first URL and continues sequentially for each additional selector. STATUS attribute Status of the invalidation. Status is one of the following: - SUCCESS for successful invalidations - URI NOT CACHEABLE for documents that are not cacheable - URI NOT FOUND for documents not found NUMINV attribute Number of documents invalidated during the invalidation request
INFO element	Returns the comment specified in the INFO element of the invalidation request

See Also: "Invalidation Request and Response DTD" on page C-3 for further information about invalidation response syntax

Invalidation Preview Response Syntax

Invalidation preview responses for preview requests are returned in the following format:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONPREVIEWRESULT SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONPREVIEWRESULT VERSION="WCS-1.1" STATUS="status" NUMURLS="number"
TOTALNUMURLS="total_number">
   <SELECTURL VALUE="URL">
   </SELECTEDURL>
</INVALIDATIONPREVIEWRESULT>
```

The body of a valid invalidation preview response begins with the following:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONPREVIEWRESULT SYSTEM "internal:///WCSinvalidation.dtd">
```

The first line denotes version 1.0 of XML. The second line denotes the response is an invalidation preview response using the `WCSinvalidation.dtd` file as the XML document type.

Notes:

- No white space is allowed before "`<?xml`".
- If an application is sharing invalidation requests with a third-party XML parser, replace `"internal:///WCSinvalidation.dtd"` with the following path:

  `"http://www.oracle.com/webcache/90200/WCSinvalidation.dtd"`

The root element INVALIDATIONPREVIEWRESULT contains one or more of the attributes and elements described in Table 8–4. BASICSELECTOR and ADVANCEDSELECTOR are described in Table 8–1 on page 8-12.

Table 8–4 INVALIDATIONPREVIEWRESULT Elements and Attributes

Invalidation Element/Attribute	Description
VERSION attribute	Version number of the WCSinvalidation.dtd file to use as the XML document type
STATUS attribute	Status of the preview. Status can be one of the following: ■ SUCCESS for successful invalidations ■ URI NOT CACHEABLE for documents that are not cacheable ■ URI NOT FOUND for documents not found
STARTNUM attribute	Number representing the first document to be listed
NUMURLS attribute	Number of URLs returned in this preview result
TOTALNUMURLS attribute	Number of URLs matching the BASICSELECTOR or ADVANCEDSELECTOR selectors
SELECTEDURL element	URLs matching the BASICSELECTOR or ADVANCEDSELECTOR selectors to be invalidated

Manual Invalidation Using Oracle9*i*AS Web Cache Manager

Oracle9*i*AS 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 Oracle9*i*AS 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 Oracle9*i*AS Web Cache Manager:

1. Start Oracle9*i*AS Web Cache Manager.

   See Also: "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. In the navigator pane, select **Administration** > **Content Invalidation**.

   The Content Invalidation page appears in the right pane.

3. From the **For Cache** list, select a cache. (More than one cache is listed only if you configured a cache cluster.)

4. Specify which documents to invalidate:

> **Note:** When using Oracle9*i*AS 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 documents to be invalidated. Include the complete path and file name.

Advanced Invalidation

URL Path Prefix	***Required.*** Specify the prefix path of the documents to be invalidated. The prefix path must begin with `http|https://`*`host_name`*`:`*`port/path/filename`* or with "`/`" and end with "`/`". *`host_name`*`:`*`port`* is optional. You can also specify the site host name and port in the **Host Name** field. 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 documents 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 (\). **Note:** The request URL that browsers send to Oracle9*i*AS Web Cache and the URL that Oracle9*i*AS Web Cache uses internally for that request are different. When Oracle9*i*AS Web Cache serves a page request, it alphabetically sorts any embedded URL parameters of the URL. However, the invalidation requests are matched against only the internal representation of the URL in which any embedded URL parameters are sorted. To ensure invalidation requests are matched correctly, sort and enter the embedded URL parameters alphabetically.
Host Name	***Optional.*** Specify the host name and port number of the site (`host_name:port`). Port 80 is the default port for HTTP and port 443 is the default port for HTTPS. This field is required if the **URL Path Prefix** does not include `http:https://host_name:port/path/filename`.
HTTP Method	***Optional.*** Select the HTTP request method (`GET` or `POST`) of the documents to be invalidated.
POST Body Expression	***Optional.*** If **POST** is selected for the **HTTP Method**, enter the HTTP body of the documents to be invalidated.
Cookie Information	***Optional.*** Enter the cookie name for the documents to be invalidated in the **Name** field, and enter its value in the **Value** field.
Header Information	***Optional.*** Enter the HTTP request header for the documents 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.

5. Optionally, you can preview the list of documents to be invalidated to ensure that you are removing only the documents you want to remove.

   To preview the list of documents:

   a. In the **Action** section, choose **Preview list of documents to be removed**.

   b. Specify the **Document Range**:

Start number	Enter the number representing the first document to be listed. Oracle9*i*AS Web Cache begins the count of documents with the number 0.
Maximum number of documents to be listed	Enter the number of documents to be listed. If fewer documents than the number specified meet the invalidation criteria, then Oracle9*i*AS Web Cache lists the URLs for only those documents that meet the criteria. If more documents than the number specified meet the invalidation criteria, then Oracle9*i*AS Web Cache lists the URLs for the number of documents requested. It also returns the total number of documents that meet the invalidation criteria.

   c. Choose **Submit.**

      Oracle9*i*AS Web Cache displays the Invalidation Preview Results message box, which lists the documents that meet the invalidation criteria. Oracle9*i*AS Web Cache Manager lists only those documents that are valid. Although the cache may contain documents that are expired or that have been invalidated, those documents are not listed.

      If the documents listed are those that you want to invalidate, then continue with the next step. If they are not, then modify the invalidation criteria and preview the list again.

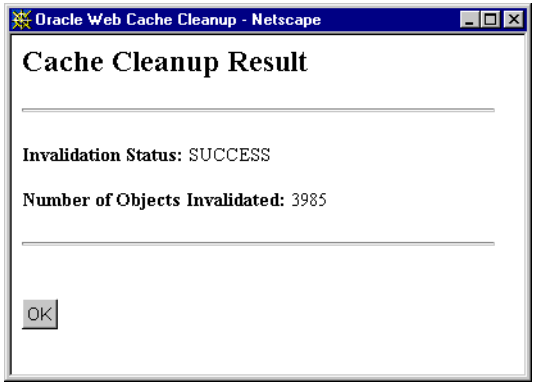
6. In the **Action** section, specify how to process invalid documents.

Remove immediately	Select this option to have Oracle9*i*AS 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 Oracle9*i*AS 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.

7. Choose **Submit**.

Oracle9*i*AS Web Cache processes the invalidation request, and returns a Cache Cleanup dialog box that the shows the invalidation status and the number of objects invalidated. For example:



For prefix-based invalidations that require Oracle9*i*AS 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 Oracle9*i*AS Web Cache.

In a cache cluster environment, if **Propagate Invalidation** is enabled, Oracle9*i*AS Web Cache sends the invalidation requests to one cluster member who acts as the **invalidation coordinator**. The coordinator propagates the invalidation requests to other cluster members. When the invalidation has been completed for all cluster members, Oracle9*i*AS Web Cache returns a Cache Cleanup box, that lists, for each cluster member, the cache name, the status of the invalidation request, and the number of objects invalidated.

See Also:

- "Task 1: Configure the Cache Cluster Settings" on page 6-56 for information about enabling invalidation propagation
- "Invalidation in Cache Clusters" on page 10 for further information about invalidation propagation in a cache cluster

Automatic Invalidation Using Applications

Invalidation requests can originate from a Web site's underlying application logic or from the content management application used to design Web pages.

Oracle9*i*AS Web Cache ships with the following Application Program Interfaces (APIs) that you can implement:

- `jawc.jar` for a Java invalidation API

  `jawc.jar` is located in the `$ORACLE_HOME/webcache/jlib` directory on UNIX and *`ORACLE_HOME`*`\webcache\jlib` directory on Windows

- `wxvutil.sql` and `wxvappl.sql` for a PL/SQL invalidation API

  `wxvutil.sql` and `wxvappl.sql` are located in the `$ORACLE_HOME/webcache/examples` directory on UNIX and *`ORACLE_HOME`*`\webcache\examples` directory on Windows

  See Also:

  - *Invalidation API Reference (Javadoc)* in the documentation library
  - `readme.examples.html` in the `$ORACLE_HOME/webcache/examples` directory on UNIX and *`ORACLE_HOME`*`\webcache\examples` directory on Windows for further information about `wxvutil.sql` and `wxvappl.sql`

Oracle9*i*AS Web Cache also ships with the following sample code for generating invalidation requests. You can create invalidation tools following these examples and use them with your applications.

- `Invalidate.java` for Java source
- `invalidate.c` for C source
- `invalidate.sh` for a shell script that invalidates documents with a `telnet` session
- `invalidate.sql` for PL/SQL source

These files are located in the `$ORACLE_HOME/webcache/examples` directory on UNIX and *`ORACLE_HOME`*`\webcache\examples` directory on Windows.

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 these files

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 requests. To do this, use the `UTL_TCP` Oracle supplied package to send invalidation requests 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 Oracle9*i*AS 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 request to the cache invalidating all catalog views and search results. (Note that the invalidation request need not list every individual search page or item view that might be effected by the data change.) The performance assurance feature of Oracle9*i*AS 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.

Invalidation Examples

This section contains the following invalidation request 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
- Example: Invalidating Documents Using Substring and Query String Matching
- Example: Propagating Invalidation Requests Throughout a Cache Cluster
- Example: Previewing Invalidation

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 request or `BODY` information.

Example: Invalidating One Document

The following request invalidates the file `/images/logo.gif`:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.1">
   <OBJECT>
     <BASICSELECTOR URI="http://www.company.com:80/images/logo.gif"/>
     <ACTION/>
   </OBJECT>
</INVALIDATION>
```

Invalidation response:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONRESULT VERSION="WCS-1.1">
   <OBJECTRESULT>
     <BASICSELECTOR URI="http://www.company.com:80/images/logo.gif"/>
     <RESULT ID="1" STATUS="SUCCESS" NUMINV="1"/>
     </OBJECTRESULT>
</INVALIDATIONRESULT>
```

The following request 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-1.1">
  <OBJECT>
    <BASICSELECTOR URI="http://www.company.com:80/contacts/contacts.html"/>
    <ACTION REMOVALTTL="0"/>
  </OBJECT>
</INVALIDATION>
```

This is equivalent to the following request using the ADVANCEDSELECTOR element. This request specifies the site information in the HOST attribute.

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.1">
  <OBJECT>
    <ADVANCEDSELECTOR URIPREFIX="/contacts/" URIEXP="^/contacts/contacts\.html$"
HOST="www.company.com:80"/>
    <ACTION REMOVALTTL="0"/>
  </OBJECT>
</INVALIDATION>
```

The second request specifies the site information in the URIPREFIX attribute

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.1">
  <OBJECT>
    <ADVANCEDSELECTOR URIPREFIX="http://www.company.com/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 request 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 request invalidates two different objects, `summary.jsp` and `summary.gif`. In addition, the request provides the comments `"summary.jsp"` and `"summary.jsp"` to be included in the invalidation result and event log.

```
<?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"
HOST="www.company.com:80"/>
      <COOKIE NAME="group" VALUE="asia"/>
    </ADVANCEDSELECTOR>
    <ACTION />
    <INFO VALUE="summary.jsp"/>
  </OBJECT>
  <OBJECT>
    <ADVANCEDSELECTOR URIPREFIX="/image/" URIEXP="summary.*\.gif$"
HOST="www.company.com:80"/>
    <INFO VALUE="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"
HOST="www.company.com:80"/>
      <COOKIE  NAME="group"  VALUE="asia"  />
    </ADVANCEDSELECTOR>
    <RESULT ID="1" STATUS="SUCCESS" NUMINV="2"/>
    <INFO VALUE="summary.jsp"/>
  </OBJECTRESULT>
  <OBJECTRESULT>
    <ADVANCEDSELECTOR URIPREFIX="/image/" URIEXP="summary.*\.gif$"
HOST="www.company.com:80"/>
    </ADVANCEDSELECTOR>
    <RESULT ID="2" STATUS="SUCCESS" NUMINV="14"/>
    <INFO VALUE="summary.gif"/>
  </OBJECTRESULT>
</INVALIDATIONRESULT>
```

The following messages are written to the event log:

```
01/Oct/2001:23:51:48 +0000 -- Information: <Invalidation>Invalidation with info
'summary.jsp' has returned with status 'SUCCESS'; number of documents
invalidated: '2'.
.
.
.
01/Oct/2001:23:51:48 +0000 -- Information: <Invalidation>Invalidation with info
'summary.gif' has returned with status 'SUCCESS'; number of documents
invalidated: '14'.
```

Example: Invalidating a Subtree of Documents

The following request 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/" HOST="www.company.com:80"/>
     <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/" HOST="www.company.com:80"/>
     <RESULT ID="1" STATUS="SUCCESS" NUMINV="125"/>
   </OBJECTRESULT>
</INVALIDATIONRESULT>
```

The following request invalidates all documents under the `/contacts/` directory whose file names end 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$"
HOST="www.company.com:80"/>
      <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$"
HOST="www.company.com:80"/>
      <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 request invalidates all documents under /.

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.0">
   <OBJECT>
     <ADVANCEDSELECTOR URIPREFIX="/" HOST="www.company.com:80"/>
     <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="/" HOST="www.company.com:80"/>
      <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.

The following syntax invalidates sample.gif files within the /cec/cstage/graphic* directories:

```
<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.

The following syntax instructs Oracle9*i*AS Web Cache to locate a directory named graphic*:

```
<ADVANCEDSELECTOR URIPREFIX="/cec/cstage/graphic*/" URIEXP="sample\.gif"
HOST="www.company.com:80"/>
</ADVANCEDSELECTOR>
```

The following syntax invalidates documents contained within /cec/cstage?ecaction=viewitem:

```
<ADVANCEDSELECTOR URIPREFIX="/cec/" URIEXP="cstage\?ecaction=viewitem"
HOST="www.company.com:80"/>
</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.

Example: Invalidating Documents Using Substring and Query String Matching

The following request invalidates all documents under / matching the substrings /post/ and htm. In addition, the request provides the comment "remove-htm-under-all-post-dir" to be included in the invalidation result and event log.

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.1">
  <OBJECT>
    <ADVANCEDSELECTOR URIPREFIX="/"
                      HOST="www.company.com:80">
      <OTHER NAME="URI" TYPE="SUBSTRING" VALUE="/post/"/>
      <OTHER NAME="URI" TYPE="SUBSTRING" VALUE="htm"/>
    </ADVANCEDSELECTOR>
    <ACTION REMOVALTTL="0" />
    <INFO VALUE="remove-htm-under-all-post-dir"/>
  </OBJECT>
</INVALIDATION>
```

Invalidation response:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONRESULT VERSION="WCS-1.0">
   <OBJECTRESULT>
     <ADVANCEDSELECTOR URIPREFIX="/" HOST="www.company.com:80"/>
       <OTHER NAME="URI" TYPE="SUBSTRING" VALUE="/post/"/>
       <OTHER NAME="URI" TYPE="SUBSTRING" VALUE="htm"/>
      <RESULT ID="1" STATUS="SUCCESS" NUMINV="52"/>
      <INFO VALUE="remove-htm-under-all-post-dir"/>
   </OBJECTRESULT>
</INVALIDATIONRESULT>
```

The following message is written to the event log:

```
01/Oct/2001:23:51:48 +0000 -- Information: <Invalidation>Invalidation with info
'remove-htm-under-all-post-dir' has returned with status 'SUCCESS'; number of
documents invalidated: '52'.
```

The following request invalidates all documents under `/corporate/asp/`, matching the substring `/view_building.asp/` and the embedded URL parameter value pairs of `building=8` and `floor=10`. In addition, the request provides the comment `"remove-view-building8-10th-floor"` to be included in the invalidation result and event log.

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.1">
  <OBJECT>
    <ADVANCEDSELECTOR URIPREFIX="/corporate/asp/"
                      HOST="www.company.com:80">
      <OTHER NAME="URI" TYPE="SUBSTRING" VALUE="/view_building.asp"/>
      <OTHER NAME="QUERYSTRING_PARAMETER" TYPE="SUBSTRING" VALUE="building=8"/>
      <OTHER NAME="QUERYSTRING_PARAMETER" TYPE="SUBSTRING" VALUE="floor=10"/>
    </ADVANCEDSELECTOR>
    <ACTION REMOVALTTL="0" />
    <INFO VALUE="remove-view-building8-10th-floor"/>
  </OBJECT>
</INVALIDATION>
```

Invalidation response:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONRESULT VERSION="WCS-1.0">
   <OBJECTRESULT>
     <ADVANCEDSELECTOR URIPREFIX="/" HOST="www.company.com:80"/>
       <OTHER NAME="URI" TYPE="SUBSTRING" VALUE="/view_building.asp"/>
       <OTHER NAME="QUERYSTRING_PARAMETER" TYPE="SUBSTRING" VALUE="building=8"/>
      <OTHER NAME="QUERYSTRING_PARAMETER" TYPE="SUBSTRING" VALUE="floor=10"/>
       <RESULT ID="1" STATUS="SUCCESS" NUMINV="3"/>
       <INFO VALUE="remove-view-building8-10th-floor"/>
   </OBJECTRESULT>
</INVALIDATIONRESULT>
```

The following message is written to the event log:

```
01/Oct/2001:23:51:48 +0000 -- Information: <Invalidation>Invalidation with info
'remove-view-building8-10th-floor' has returned with status 'SUCCESS'; number of
documents invalidated: '3'.
```

> **See Also:** "Enhance Query String Invalidations" on page 8-47 to optimize invalidations using `QUERYSTRING_PARAMETER`

Example: Propagating Invalidation Requests Throughout a Cache Cluster

In a cache cluster, you can enable or disable the propagation of invalidation requests to all cluster members. You specify the setting on the Cluster Configuration page (**Administration** > **Cluster Configuration**) of Oracle9*i*AS Web Cache Manager.

You can override the setting by using a pair of name/value attributes of the SYSTEMINFO element. If NAME is set to WCS_PROPAGATE and VALUE is set to TRUE, it overrides the setting specified in Oracle9*i*AS Web Cache Manager. If NAME is set to WCS_PROPAGATE and VALUE is set to FALSE, then it reads the setting specified in Oracle9*i*AS Web Cache Manager.

The following request invalidates the file `/images/logo.gif` and propagates the request to all cluster members. In this example, there are three cluster members:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.1">
   <SYSTEM>
      <SYSTEMINFO NAME=″WCS_PROPAGATE″ VALUE=″TRUE″/>
   </SYSTEM>
   <OBJECT>
     <BASICSELECTOR URI="/hostname:port/images/logo.gif"/>
     <ACTION/>
   </OBJECT>
</INVALIDATION>
```

Invalidation response:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULTDETAIL SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONRESULTDETAIL VERSION="WCS-1.1">
  <INVALIDATIONRESULT VERSION="WCS-1.1">
     <SYSTEM>
       <SYSTEMINFO NAME=″WCS_CACHE_NAME″ VALUE=″Cache_A″/>
     </SYSTEM>
     <OBJECTRESULT>
       <BASICSELECTOR URI="http://www.company.com:80/images/logo.gif"/>
       <RESULT ID="1" STATUS="SUCCESS" NUMINV="1"/>
     </OBJECTRESULT>
  </INVALIDATIONRESULT>
  <INVALIDATIONRESULT VERSION="WCS-1.1">
     <SYSTEM>
       <SYSTEMINFO NAME=″WCS_CACHE_NAME″ VALUE=″Cache_B″/>
     </SYSTEM>
     <OBJECTRESULT>
       <BASICSELECTOR URI="http://www.company.com:80/images/logo.gif"/>
       <RESULT ID="1" STATUS="SUCCESS" NUMINV="1"/>
     </OBJECTRESULT>
  </INVALIDATIONRESULT>
  <INVALIDATIONRESULT VERSION="WCS-1.1">
     <SYSTEM>
       <SYSTEMINFO NAME=″WCS_CACHE_NAME″ VALUE=″Cache_C″/>
     </SYSTEM>
     <OBJECTRESULT>
       <BASICSELECTOR URI="http://www.company.com:80/images/logo.gif"/>
       <RESULT ID="1" STATUS="SUCCESS" NUMINV="1"/>
     </OBJECTRESULT>
  </INVALIDATIONRESULT>
</INVALIDATIONRESULTDETAIL>
```

Example: Previewing Invalidation

The following request previews up to 50 documents ending in `*.htm`:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONPREVIEW SYSTEM
"internal:///WCSinvalidation.dtd">
<INVALIDATIONPREVIEW VERSION="WCS-1.1" STARTNUM="0" MAXNUM="50">
  <ADVANCEDSELECTOR URIPREFIX="http://company-sun/"
                    URIEXP=".*\.htm" >
  </ADVANCEDSELECTOR>
</INVALIDATIONPREVIEW>
```

Invalidation response:

```
"<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONPREVIEWRESULT SYSTEM
"internal:///WCSinvalidation.dtd">
<INVALIDATIONPREVIEWRESULT VERSION="WCS-1.1" STATUS="SUCCESS"
                           STARTNUM="0" NUMURLS="2" TOTALNUMURLS="2">
  <SYSTEM>
    <SYSTEMINFO NAME="WCS_CACHE_NAME" VALUE="server-cache"/>
  </SYSTEM>
  <SELECTEDURL VALUE="/company-sun:80/index.htm "/>
  <SELECTEDURL VALUE="/company-sun:80/dtd.htm "/>
</INVALIDATIONPREVIEWRESULT>
```

Reducing Invalidation Overhead

When Oracle9*i*AS Web Cache receives an advanced invalidation request, it traverses the contents of the cache to locate the objects to invalid. Depending on the structure and number of objects cached, it can take time for Oracle9*i*AS Web Cache to invalid content. Here are some ways you can expedite cache content traversal:

- Send Basic Invalidation Requests for Invalidating One Object
- Use Substring Matching for Invalidating Multiple Objects in Advanced Invalidations
- Enhance Query String Invalidations

Send Basic Invalidation Requests for Invalidating One Object

When you need to invalidate one object in the cache, send a basic rather than an advanced invalidation request to avoid cache traversal.

To send a basic invalidation request, use the **Basic Invalidation** option in the Content Invalidation page (**Administration** > **Content Invalidation**) in Oracle9*i*AS Web Cache Manager or specify the `BASICSELECTOR` element in a manual invalidation request. For example, the following request 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-1.1">
   <OBJECT>
     <BASICSELECTOR URI="http://www.company.com:80/contacts/contacts.html"/>
     <ACTION REMOVALTTL="0"/>
   </OBJECT>
</INVALIDATION>
```

Advanced invalidation requests should be reserved for invalidation of multiple objects. To send an advanced invalidation request, use the **Advanced Invalidation** option in the Content Invalidation page or specify the `ADVANCEDSELECTOR` element in a manual invalidation request.

Use Substring Matching for Invalidating Multiple Objects in Advanced Invalidations

When multiple objects share a common URL, request POST body, or an embedded URL parameter, you can express the common elements in multiple ways:

Common URL:

- Use the **URL Regular Expression** field in the Content Invalidation page (**Administration** > **Content Invalidation**) in Oracle9*i*AS Web Cache Manager.
- Use the URIEXP attribute of the ADVANCEDSELECTOR element.
- Use the OTHER element of the ADVANCEDSELECTOR element with a NAME attribute value of URI.

Common Request POST Body:

- Use the HTTP Method and POST Body Expression fields in the Content Invalidation page
- Use the METHOD and BODYEXP attributes of the ADVANCEDSELECTOR element.
- Use the OTHER element of the ADVANCEDSELECTOR element with a NAME attribute value of BODY.

Common Embedded URL Parameter:

- Use the OTHER element of the ADVANCEDSELECTOR element with a NAME attribute value of QUERYSTRING_PARAMETER.

For the quickest invalidation, Oracle Corporation recommends using the OTHER element to specify a substring for literal matching rather than regular expression for pattern matching.

To send an advanced invalidation request with substring matching,

1. Specify the OTHER element in a manual invalidation request that uses the ADVANCEDSELECTOR element.
2. Specify the NAME attribute to use a value of URI, BODY, or QUERYSTRING_PARAMETER.
3. Specify the TYPE attribute to use a value of SUBSTRING.

For example, the following request searches for any documents underneath `http://wc-cluster.us.oracle.com:1100/pls/portal/!PORTAL.wwpro_app_provider.execute_portlet/595897563/`, that match the following criteria:

- The HTTP request method is an HTTP `POST` request method
- The `URI` is `showPortlet.Show`
- The HTTP `POST` body contains `_language=EN-US`
- The HTTP requests headers are `x-oracle-cache-user` and `x-oracle-cache-subid`
- The embedded URL parameters are `_portlet_id` and `_provider_id`

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATION SYSTEM
"http://www.oracle.com/webcache/90200/WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.1">
  <OBJECT>
    <ADVANCEDSELECTOR
URIPREFIX="/pls/portal/!PORTAL.wwpro_app_provider.execute_portlet/595897563/"
HOST="wc-cluster.us.oracle.com:1100" METHOD="POST">
      <OTHER NAME="QUERYSTRING_PARAMETER" TYPE="SUBSTRING"
VALUE="_portlet_id=2"/>
      <OTHER NAME="QUERYSTRING_PARAMETER" TYPE="SUBSTRING"
VALUE="_provider_id=595897563"/>
      <HEADER NAME="x-oracle-cache-user" VALUE="PORTAL"/>
      <HEADER NAME="x-oracle-cache-subid" VALUE="1"/>
      <OTHER NAME="BODY" TYPE="SUBSTRING" VALUE="_language=EN-US"/>
      <OTHER NAME="URI" TYPE="SUBSTRING" VALUE="showPortlet.Show"/>
    </ADVANCEDSELECTOR>
    <ACTION REMOVALTTL="0"/>
    <INFO VALUE="Invalidate an old portlet based on portlet ID and
provider ID"/>
  </OBJECT>
</INVALIDATION>
```

Enhance Query String Invalidations

If you are using the QUERYSTRING_PARMETER in an invalidation request, it can take additional processing time Oracle9*i*AS Web Cache to match and invalidate the documents.

To optimize the invalidation process for query string invalidations:

1. Use a text editor to open the webcache.xml file.
2. Locate the SECURITY and WATCHDOG elements:

```
<?xml version="1.0"?>
<!DOCTYPE CALYPSO SYSTEM "internal:///webcache.dtd">
<CALYPSO>
<VERSION DTD_VERSION="2.0.4"/>
...
  <SECURITY>
    <USER .../>
    <USER .../>
    <SECURESUBNET ...>
    </SECURESUBNET>
    <DEBUGINFO HEADER="YES" BODY="NO"/>
  </SECURITY>

  <WATCHDOG ENABLE="YES"/>
...
```

3. Between the SECURITY and WATCHDOG elements, add an INVALIDATIONINDEX element in the following format:

```
  <SECURITY>
  ...
  </SECURITY>

  <INVALIDATIONINDEX>
   <INDEXPARAM VALUE="VALUE_of_QUERYSTRING_PARAMETER"/>
   <INDEXPARAM VALUE="VALUE_of_QUERYSTRING_PARAMETER"/>
  </INVALIDATIONINDEX>

  <WATCHDOG ENABLE="YES|NO"/>
```

4. Restart Oracle9*i*AS Web Cache with the webcachectl restart command.

The following example request invalidates all documents under `/corporate/asp/` matching the substring `/view_building.asp` and the embedded URL parameter value pairs of `building=8` and `floor=10`. In addition, the request provides the comment `"remove-view-building8-10th-floor"` to be included in the invalidation result and event log.

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.1">
  <OBJECT>
    <ADVANCEDSELECTOR URIPREFIX="/corporate/asp/"
                      HOST="www.company.com:80">
      <OTHER NAME="URI" TYPE="SUBSTRING" VALUE="/view_building.asp"/>
      <OTHER NAME="QUERYSTRING_PARAMETER" TYPE="SUBSTRING" VALUE="building=8"/>
      <OTHER NAME="QUERYSTRING_PARAMETER" TYPE="SUBSTRING" VALUE="floor=10"/>
    </ADVANCEDSELECTOR>
    <ACTION REMOVALTTL="0" />
    <INFO VALUE="remove-view-building8-10th-floor"/>
  </OBJECT>
</INVALIDATION>
```

To optimize this search, update `webcache.xml` file with the following:

```
<?xml version="1.0"?>
<!DOCTYPE CALYPSO SYSTEM "internal:///webcache.dtd">
<CALYPSO>
<VERSION DTD_VERSION="2.0.4"/>
...
  <SECURITY>
    <USER TYPE="INVALIDATION" PASSWORDHASH="48690"/>
    <USER TYPE="MONITORING" PASSWORDHASH="48690"/>
    <SECURESUBNET ALLOW="ALL">
      <IP ADDR="1.2.3.4900000"/>
      <IP ADDR="5.6.7.8.9999"/>
    </SECURESUBNET>
  </SECURITY>

  <INVALIDATIONINDEX>
    <INDEXPARAM VALUE="building"/>
    <INDEXPARAM VALUE="floor"/>
  </INVALIDATIONINDEX>

  <WATCHDOG ENABLE="YES"/>
...
```

See Also:

- Table 8–1, " INVALIDATION Elements and Attributes" on page 8-12 for a description of the invalidation request syntax
- "Example: Invalidating Documents Using Substring and Query String Matching" on page 8-39

Listing the Contents of the Cache

With Oracle9*i*AS Web Cache Manager, you can list the contents of the cache, generating the following types of lists:

- A list of the URLs of the most popular documents stored in the cache since the cache was last started

  You specify the number of objects and Oracle9*i*AS Web Cache Manager displays the list in the Cache Contents page.

- A list of the URLs of the objects currently in the cache

  You specify a file name and Oracle9*i*AS Web Cache writes the list of URLs to the file. You can use this list to verify that the cacheability rules are caching the objects that you want cached.

Note: Oracle9*i*AS Web Cache Manager lists only those documents that are valid. Although the cache may contain documents that are expired or that have been invalidated, those documents are not included in these lists.

Listing Popular Documents

To view the list of URLs of the most popular documents, perform following steps:

1. Start Oracle9*i*AS Web Cache Manager.

   See Also: "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. In the navigation pane, select **Administration** > **Cache Contents**.

   The Cache Contents page appears in the right pane.

3. From the **For Cache** list, select a cache. (More than one cache is listed only if you configured a cache cluster.)

4. For **List Most Popular Objects in Cache**, enter the number of URLs to display in the **Number of Objects** field.
5. Choose **Update**.

   Oracle9*i*AS Web Cache Manager displays a table containing the list of URLs of the most popular objects since the cache was last started. The table contains the following columns:

Rank	A ranking, from 1 to 100, based on the score of the object. A rank of 1 represents the object with the highest score; that is, the most popular object.
Object Name	The URL of the object. The URLs may contain additional descriptive information, such as cookie or session information.
Size	The size of the object. The size is represented in bytes, kilobytes (KB), or megabytes (MB).
Cacheability Rule	If there is a cacheability rule associated with the object, then this column displays a link to the **Cacheability Rule Details** page that shows the regular expression and site information for the URL.

Listing All Contents

You can also generate a list of all of the objects in the cache. However, to maintain the performance of the cache, Oracle Corporation recommends that you perform this operation during non-peak hours. While writing the list of URLs to the text file, performance may degrade somewhat.

To generate a list of the URLs of all of the documents currently in the cache:

1. Start Oracle9*i*AS Web Cache Manager.

   See Also: "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. In the navigation pane, select **Administration** > **Cache Contents**.

   The Cache Contents page appears in the right pane.
3. From the **For Cache** list, select a cache. (More than one cache is listed only if you have a cache cluster.)
4. For **List All Contents in Cache**, choose **Export to File**.

The Export Cache Contents dialog box appears. It lists the file to which it will write the URLs. By default, the file is written to the Oracle9*i*AS Web Cache log directory and is named `webcache_contents.txt`.

5. To write the list to a different location, enter a complete file specification in the text box.

6. Choose **Submit.**

   Oracle9*i*AS Web Cache writes the list of URLs to the text file you specified. Each time you generate the list, Oracle9*i*AS Web Cache appends the data to the existing file. It lists the date that the data was appended to the file, followed by the URLs of the objects currently cached. The following example shows an excerpt of the `webcache_contents.txt` file:

```
Cache Contents at Wed Aug 22 11:47:03 2001
www.company.com:80/images/lnav/lnav_products.gif
www.company.com:80/images/rnav/rnav_red_line_1.gif
www.company.com:80/images/bullets_and_symbols/blk_line_bullet_10.gif
.
.
.
Cache Contents at Wed Aug 22 13:01:24 2001
www.company.com:80/images/white_spacer_xp.gif
www.company.com:80/images/white_spacer.gif
www.company.com:80/images/miniappsnet.gif
.
.
.
```

Evaluating Event Logs

Oracle9*i*AS Web Cache events and errors are stored in an **event log**. The event log can help you determine which 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.

This section contains the following topics:

- Format of the Event Log File
- Event Log Examples
- Finding Errors in the Event Log
- Configuring Event Logs

> **See Also:** Appendix E 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:

```
13/Nov/2001:19:26:59 +0000 -- Information: Maximum number of file/socket
descriptors set to 910.
13/Nov/2001:19:26:59 +0000 -- Information: Maximum connections possible are 750
13/Nov/2001:19:26:59 +0000 -- Information: Listening on ADMINISTRATION port 4000
address 0.0.0.0
13/Nov/2001:19:26:59 +0000 -- Information: The admin server started successfully
13/Nov/2001:19:26:59 +0000 -- Information: Maximum number of file/socket
descriptors set to 910.
13/Nov/2001:19:26:59 +0000 -- Information: Maximum connections possible are 750
13/Nov/2001:19:27:00 +0000 -- Information: Listening on NORM port 7777 address
0.0.0.0
13/Nov/2001:19:27:00 +0000 -- Information: Listening on NORMSSLV3_V2H port 4443
address 0.0.0.0
13/Nov/2001:19:27:00 +0000 -- Information: Listening on INVALIDATION port 4001
address 0.0.0.0
13/Nov/2001:19:27:00 +0000 -- Information: Listening on STATISTICS port 4002
address 0.0.0.0
13/Nov/2001:19:27:00 +0000 -- Information: A 1 node cluster successfully
initialized
13/Nov/2001:19:27:00 +0000 -- Information: The cache server started successfully
13/Nov/2001:19:27:00 +0000 -- Information: The cache server is started by the
admin server at startup
13/Nov/2001:19:27:00 +0000 -- Information: Auto-Restart: WXE-00800 Auto-restart
started successfully
```

Example: Event Log with Unsuccessful Startup Entries

The following shows an event log excerpt with unsuccessful startup events. Oracle9*i*AS Web Cache is unable to listen on port 7777, because it is already in use. This can occur if Oracle9*i*AS Web Cache is already running and listening on that port or another application is using that port.

```
13/Nov/2001:19:34:27 +0000 -- Information: Maximum number of file/socket
descriptors set to 910.
13/Nov/2001:19:34:27 +0000 -- Information: Maximum connections possible are 750
13/Nov/2001:19:34:27 +0000 -- Information: Listening on ADMINISTRATION port 4000
address 0.0.0.0
13/Nov/2001:19:34:27 +0000 -- Information: The admin server started successfully
13/Nov/2001:19:34:28 +0000 -- Information: Maximum number of file/socket
descriptors set to 910.
13/Nov/2001:19:34:28 +0000 -- Information: Maximum connections possible are 750
13/Nov/2001:19:34:28 +0000 -- Error: Unable to listen on port 7777
13/Nov/2001:19:34:28 +0000 -- Error: Failed to start the server.
13/Nov/2001:19:34:28 +0000 -- Error: The server could not initialize
13/Nov/2001:19:34:28 +0000 -- Information: The server is exiting
13/Nov/2001:19:34:28 +0000 -- Warning: The admin server couldn't start the cache
server, running in admin-only mode.
```

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 `cache.htm`:

```
14/Nov/2001:23:58:23 +0000 -- Information: <Invalidation>1 URLs with
prefix /cache.htm have been successfully invalidated.
14/Nov/2001:23:58:23 +0000 -- Information: <Invalidation>Invalidation
with info 'remove cache static page' has returned with status
'SUCCESS'; number of documents invalidated: '1'.
```

Examples: Event Log with Invalidation Request Errors

The following shows an event log excerpt with an XML invalidation request error. In this example, Oracle9*i*AS Web Cache is unable to parse the request:

```
15/Nov/2001:00:00:11 +0000 -- Error: XML Parsing Error in NULL. Error
code 115. LPX-00115: Warning: element "BASICSELECTOR" missing required
attribute "URI"
15/Nov/2001:00:00:11 +0000 -- Error: WebCache failed to parse XML.
Error code 115
15/Nov/2001:00:00:11 +0000 -- Error: <Invalidation>Invalidation XML
Buffer cannot be parsed.
```

The following shows an event log excerpt with invalidation request for nonexisting documents:

```
15/Nov/2001:00:04:29 +0000 -- Information: <Invalidation>Requested URI
/cache.htm is not found in the cache.  URI is not invalidated.
15/Nov/2001:00:04:29 +0000 -- Information: <Invalidation>Invalidation
with info 'remove cache static page' has returned with status 'URI NOT
FOUND'; number of documents invalidated: '0'.
```

Example: Event Log with Shutdown Entries

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

```
13/Nov/2001:19:28:21 +0000 -- Information: SIGTERM caught - program will shut
down once all connections are complete.
13/Nov/2001:19:28:21 +0000 -- Information: The server is exiting
13/Nov/2001:19:28:22 +0000 -- Information: SIGTERM caught - program will shut
down once all connections are complete.
13/Nov/2001:19:28:22 +0000 -- 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:" event*
```

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

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

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

Configuring Event Logs

To establish event log configuration settings:

1. Start Oracle9*i*AS Web Cache Manager.

   See Also: "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. In the navigator pane, select **Cache-Specific Configuration**> **Event Log**.

   The Event Log page appears in the right pane.

3. In the Event Log page, select a cache and choose **Edit**.

   The Change Options for Event Logging dialog box appears.

4. In **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, Oracle9*i*AS Web Cache has no mechanism to improve the performance of the conversion process.

5. From the **Rollover Frequency** list, select how often you want to change the frequency at which Oracle9*i*AS Web Cache will save current log information to `event_log.`*`yyyymmdd`* and write new log information to `event_log`.

   If you have a high-volume site, increase the frequency.

6. In **Verbose Mode**, 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.

7. Choose **Submit**.

8. Apply changes and restart Oracle9*i*AS Web Cache:

   a. In the Oracle9*i*AS Web Cache Manager main window, choose **Apply Changes**.

   b. In the navigator pane, select **Administration** > **Operations**.

      The Operations page appears in the right pane.

   c. In the Operations page, choose **Restart** to restart Oracle9*i*AS Web Cache.

Evaluating Access Logs

Oracle9*i*AS Web Cache generates an **access log** with the information about the HTTP requests sent to Oracle9*i*AS Web Cache. 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 Oracle9*i*AS Web Cache uses buffered logging for the access log, that is, it writes to the access log after the buffer is full.

This section contains the following topics:

- Format of the Access Log Files
- Access Log Examples
- Configuring Access Logs

Format of the Access Log Files

You can configure the content of the access log files by defining the fields to appear for each HTTP request event. Table 8–5 lists the fields you can enter.

Table 8–5 User-Specified for Access Logs

Field	Description
`bytes`	Content length of the transferred document
`c-auth-id`	User name if the request contained an attempt to authenticate
`c-ip`	Browser's IP address
`clf-date`	Date of when the transaction completed. The date is displayed in the following format: *`dd/Mon/yyyy`*`:`*`hh`*`:`*`mm`*`:`*`ss`*
`cs(`*`HTTP_request_header`*`)`	HTTP request header sent from the browser to Oracle9*i*AS Web Cache. **See Also:** "cs(request_header) and sc(response_header) Access Log Fields" on page 8-60
`cs-method`	Browser-to-Oracle9*i*AS Web Cache HTTP request method
`cs-uri`	Browser-to-Oracle9*i*AS Web Cache URI
`cs-uri-query`	Browser-to-Oracle9*i*AS Web Cache query portion of URI, omitting the stem

Table 8–5 (Cont.) User-Specified for Access Logs

Field	Description
`cs-uri_stem`	Browser-to-Oracle9*i*AS Web Cache stem portion of URI, omitting the query
`date`	Date of when the transaction completed. The date is displayed in the following format: *`dd/Mon/yyyy`*
`req-line`	HTTP request method, URI of the request, and HTTP version
`s-ip`	Oracle9*i*AS Web Cache's IP address
`sc(`*`HTTP_response_header`*`)`	HTTP response header sent from Oracle9*i*AS Web Cache to the browser **See Also:** "cs(request_header) and sc(response_header) Access Log Fields" on page 8-60
`sc-status`	Oracle9*i*AS Web Cache-to-browser HTTP status code: ■ 1*xx* range messages are informational. ■ 2*xx* range messages indicate success. ■ 3*xx* range messages indicate redirection, that is, further action must be taken in order to complete the request ■ 4*xx* range messages indicate a client error. ■ 5*xx* range messages indicate a Oracle9*i*AS Web Cache error **See Also:** `http://www.ietf.org/rfc/rfc2616.txt` for further information about HTTP status codes
`time`	Time at which the transaction completed. The time is displayed in the following format: *`hh`*`:`*`mm`*`:`*`ss`*
`time-end`	Time at which the transaction ended. The time is displayed in the following format: "*`seconds microseconds`*"
`time-start`	Time at which the transaction started. The time is displayed in the following format: "*`seconds microseconds`*"
`time-taken`	Amount of time taken, in microseconds, for the transaction to complete

Usage Notes

- Separate fields by a space
- The order in which fields are entered determines the order in which the fields are logged
- If no fields are specified, then the following **Common LogFile Format (CLF)** fields are used by default:

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

  Entered fields are not added to the CLF default. You must either enter no fields to use the default or enter the fields specified in Table 8–5.

cs(*request_header*) and sc(*response_header*) Access Log Fields

Table 8–6 lists examples of HTTP / 1.1 headers that can be used for the `cs(`*`request_header`*`)` and `sc(`*`response_header`*`)` fields. This table lists only some of the possible headers. It is not an exhaustive list.

Table 8–6 Examples of HTTP/1.1 Header Fields

cs(*request_header*) Field	sc(*response_header*) Field
Accept	Cache-Control
Authorization	Content-Encoding
Connection	Content-Language
Date	Content-Length
Host	Content-Type
Referer	Date
Cache-Control	ETag
Content-Encoding	Expires
Content-Language	Last-Modified
Content-Length	Pragma
Content-Type	Server
If-None-Match	Transfer-Encoding
If-Modified-Since	Via
Last-Modified	

Table 8–6 (Cont.) Examples of HTTP/1.1 Header Fields

cs(*request_header*) Field	sc(*response_header*) Field
Pragma	
Range	
TE	
User-Agent	
Via	

Table 8–7 lists examples of cookie-related headers that can be used for the cs(*request_header*) and sc(*response_header*) fields.

Table 8–7 Supported Cookie-Related Header Fields

cs(*request_header*) Field	sc(*response_header*) Field
Cookie	Set-Cookie

Table 8–8 lists examples of Oracle9*i*AS Web Cache headers that can be used for the cs(*request_header*) and sc(*response_header*) fields.

Table 8–8 Supported Oracle9iAS Web Cache Header Fields

cs(*request_header*) Field	sc(*response_header*) Field
Surrogate-Capability	Surrogate-Control

Access Log Examples

The access logs that follow uses default fields c-ip - c-auth-id [clf-date] "request line" sc-status bytes.

In the first line of the first output:

- 138.2.213.146 is the browser's IP address (c-ip)
- [19/Nov/2001:10:27:42 -0500] is the date ([clf-date])
- "GET /~ssandrew/personal.htm HTTP/1.0" is the request line ("request line")
- 200 is the HTTP status code (sc-status)
- 2438 is the size of the document sent (bytes)

```
138.2.213.146 - - [19/Nov/2001:10:27:42 -0500] "GET /~ssandrew/personal.htm
HTTP/1.0" 200 2438
138.2.213.146 - - [19/Nov/2001:10:27:54 -0500] "GET
/~ssandrew/personal.htm?UserName=Bob HTTP/1.0"
200 2438
138.2.213.146 - - [19/Nov/2001:10:47:30 -0500] "GET /~ssandrew/count.sh
HTTP/1.0" 403 289
138.2.213.146 - - [19/Nov/2001:10:47:34 -0500] "GET /~ssandrew/sbin/count.sh
HTTP/1.0" 200 321
138.2.213.146 - - [19/Nov/2001:10:47:41 -0500] "GET /sbin/count.sh HTTP/1.0" 200
321
138.2.213.146 - - [19/Nov/2001:11:34:23 -0500] "GET /cache.htm HTTP/1.0" 200 250
138.2.213.146 - - [19/Nov/2001:11:38:23 -0500] "GET /cache.htm HTTP/1.0" 304 0
138.2.213.146 - - [19/Nov/2001:11:38:48 -0500] "GET /cache.htm HTTP/1.0" 304 0
206.223.27.37 - - [19/Nov/2001:15:14:29 -0500] "GET
/~ssandrew/personal.htm?UserName=Joe HTTP/1.0"
200 2438
206.223.27.37 - - [19/Nov/2001:15:17:12 -0500] "GET
/~ssandrew/personal.htm?UserName=Shehzaad
 HTTP/1.0" 200 438
144.25.223.39 - - [19/Nov/2001:15:30:34 -0500] "GET /htdocs/coelist.html
HTTP/1.0" 200 4219
144.25.223.39 - - [19/Nov/2001:15:30:34 -0500] "GET /images/redheaderbanner.gif
HTTP/1.0" 200 1226
138.2.213.146 - - [19/Nov/2001:10:49:44 -0500] "GET /pls/coe/find_via_post
HTTP/1.0" 200 1119
138.2.213.146 - - [19/Nov/2001:10:49:44 -0500] "GET /ows-img/chalk.jpg HTTP/1.0"
404 284
130.35.35.21 - - [20/Nov/2001:00:36:35 -0500] "GET /images/support.jpg HTTP/1.0"
206 3106
130.35.35.21 - - [20/Nov/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/Nov/2001:11:04:24 -0500] "GET /cache.htm HTTP/1.0" 200 250
138.2.213.146 - - [19/Nov/2001:11:04:26 -0500] "GET /cache.htm HTTP/1.0" 200 250
138.2.213.146 - - [19/Nov/2001:11:29:24 -0500] "GET /cache.htm HTTP/1.0" 304 0
138.2.213.146 - - [19/Nov/2001:11:29:25 -0500] "GET /cache.htm HTTP/1.0" 304 0
138.2.213.146 - - [19/Nov/2001:11:29:30 -0500] "GET /cache.htm HTTP/1.0" 200 250
138.2.213.146 - - [19/Nov/2001:11:29:35 -0500] "GET /cache.htm HTTP/1.0" 200 250
```

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

Example: Access Log with Status Code 404 Entry

The following shows an access log excerpt in which Oracle9*i*AS Web Cache cannot find any objects matching the requested URL `/ows-img/chalk.jpg`. This error is indicated by HTTP status code 404.

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

Example: Access Log Host Name

The following shows an access log excerpt in which the following fields are specified:

```
c-ip c-auth-id clf-date cs(Host) req-line sc-status bytes
```

`cs(Host)` displays the output of `Host` request-header field, which specifies the site information. In this examples, requests are sent to Oracle9*i*AS Web Cache for site `www.company.com:80`.

```
148.87.1.180    -       [11/Dec/2001:18:42:04 +0000]    "www.company.com:80"
GET / HTTP/1.1  200     1456
148.87.1.180    -       [11/Dec/2001:18:42:07 +0000]    "www.company.com:80"
GET / HTTP/1.1  200     1456
148.87.1.180    -       [11/Dec/2001:18:42:08 +0000]    "www.company.com:80"
GET / HTTP/1.1  200     1456
148.87.1.180    -       [11/Dec/2001:18:42:08 +0000]    "www.company.com:80"
GET /apache_pb.gif HTTP/1.1     200     2326
148.87.1.180    -       [11/Dec/2001:18:42:09 +0000]    "www.company.com:80"
GET / HTTP/1.1  200     1456
148.87.1.180    -       [11/Dec/2001:18:42:18 +0000]    "www.company.com:80"
GET / HTTP/1.1  200     1456
148.87.1.180    -       [11/Dec/2001:18:42:18 +0000]    "www.company.com:80"
GET /apache_pb.gif HTTP/1.1     200     2326
```

Configuring Access Logs

To enable access logging:

1. Start Oracle9*i*AS Web Cache Manager.

   See Also: "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. In the navigator pane, select **Cache-Specific Configuration** > **Access Log**.

   The Access Log page appears in the right pane.

3. In the Access Log 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 **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, Oracle9*i*AS 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 Oracle9*i*AS 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 **User-Specified Fields** field, enter the fields to log.

   Separate fields by a space. Do not attempt to copy and paste the default format displays in the online help into the field. If you do, then an error will display.

9. Choose **Submit**.

10. Apply changes and restart Oracle9*i*AS Web Cache:
    a. In the Oracle9*i*AS Web Cache Manager main window, choose **Apply Changes**.
    b. In the navigator pane, select **Administration** > **Operations**.

       The Operations page appears in the right pane.
    c. In the Operations page, choose **Restart** to restart Oracle9*i*AS Web Cache.

To disable access logging:

1. Start Oracle9*i*AS Web Cache Manager.

   See Also: "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. In the navigator pane, select **Cache-Specific Configuration** > **Access Log**.

   The Access Log page appears in the right pane.
3. In the Access Log 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 Oracle9*i*AS Web Cache:
    a. In the Oracle9*i*AS Web Cache Manager main window, choose **Apply Changes**.
    b. In the navigator pane, select **Administration** > **Operations**.

       The Operations page appears in the right pane.
    c. In the Operations page, choose **Restart** to restart Oracle9*i*AS Web Cache.

Note: You can integrate Oracle9*i*AS Web Cache access logs into Oracle9*i*AS Clickstream Intelligence with the Collector Agent. See *Oracle9iAS Clickstream Intelligence Administrator's Guide* for details.

9

Monitoring Performance

See Also: *Oracle9i Application Server Performance Guide* for TCP / IP performance tuning tips

This chapter describes how to gather performance statistics with Oracle9*i*AS Web Cache Manager and how to interpret them.

This chapter contains these topics:

- Monitoring Oracle9iAS Web Cache Health
- Gathering Oracle9iAS Web Cache Performance Statistics
- Gathering Origin Server Performance Statistics

Note: You can use Oracle Enterprise Manager to monitor many of the same performance statistics described in this chapter. See *Oracle9i Application Server Administrator's Guide* for further information about using **Oracle Enterprise Manager** to monitor Oracle9*i*AS Web Cache.

Monitoring Oracle9*i*AS Web Cache Health

Oracle9*i*AS Web Cache provides a health monitor that enables you to quickly access overall cache performance.

To monitor overall cache health:

1. Start Oracle9*i*AS Web Cache Manager.

   See Also: "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. In the navigator pane, select **Administration** > **Monitoring** > **Health Monitor**.

   The Health Monitor page appears in the right pane.

3. From the **For Cache** list, select the cache. If you have not configured a cache cluster, this field displays the current cache (the cache to which you are connected.)

4. From the **Auto Refresh** list, select the frequency at which you want the statistics refreshed.

   If you select **Never**, then the page will not be refreshed again. If you want the statistics refreshed now, choose **Refresh Now**.

Table 9–1 describes the statistics for this page.

Table 9–1 Oracle9iAS Web Cache Health Monitor Statistics

Statistic	Description
Current Time	The time when this page was generated
Current Web Cache Start Time	The time when Oracle9*i*AS Web Cache was started
Time Since Start	The length of time that Oracle9*i*AS Web Cache has been operating since it was started. Time is denoted in *days/hours/minutes/seconds*.

Table 9–1 (Cont.) Oracle9iAS Web Cache Health Monitor Statistics

Statistic	Description
Total Number of Requests Served by Current Web Cache	Accumulated number of requests Oracle9*i*AS Web Cache has served since it was started **See Also:** "Gathering Oracle9iAS Web Cache Performance Statistics" on page 9-4 to view detailed statistics for Oracle9*i*AS Web Cache
Requests Served by Origin Server Table	This table provides information about the number of requests served by **origin servers**. It contains the following columns: **Requests Served by Origin Servers**: Name of the origin server and the port number from which the origin server is listening for Oracle9*i*AS Web Cache requests **Proxy Server**: **YES** specifies that the server is a proxy server. **NO** specifies that the server is an application Web server. **Up/Down**: **UP** specifies that the last communication with the server was successful. **DOWN** specifies that the server is down. If this is the last server in a single or multiple server configuration, Oracle9*i*AS Web Cache continues to forward requests. If this is not the last server, no new requests will be sent to server. However, Oracle9*i*AS Web Cache will poll the server until it is back online. **Since**: How long the origin server has been up or down **Total Request Served**: Number of Web browser requests resolved by this origin server **Average Latency**: Average amount of time for the Web browser requests to be resolved **See Also:** "Gathering Origin Server Performance Statistics" on page 9-8 to view detailed statistics for origin servers
Serving Requests/Second Now	The health bar provides a graphical view of the number of Web browser requests resolved for each second by the: ▪ Documents in the cache that have expired or that have been invalidated, but have not yet been refreshed from the origin servers ▪ Documents in the cache that are still valid

Gathering Oracle9*i*AS Web Cache Performance Statistics

To monitor Oracle9*i*AS Web Cache performance:

1. Start Oracle9*i*AS Web Cache Manager.

   See Also: "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. In the navigator pane, select **Administration**> **Monitoring** > **Web Cache Statistics**.

   The Web Cache Statistics page appears.

3. From the **For Cache** list, select the cache.

   If you have not configured a cache cluster, this field displays the current cache (the cache to which you are connected.)

4. From the **For Site** list, select the Web site for which to view statistics.

5. From the **Auto Refresh** list, select the frequency at which you want the statistics refreshed.

   If you select **Never**, then the page will not be refreshed again. If you want the statistics refreshed now, choose **Refresh Now**.

Table 9–2 describes the statistics for this page.

Table 9–2 Oracle9iAS Web Cache Statistics

Statistic	Description
Updated at	The time when this page was generated
Current Cache Start Time	The time when Oracle9*i*AS Web Cache was started or restarted
Time Since Start	The length of time that Oracle9*i*AS Web Cache has been operating since it was started or restarted. Time is denoted in *days/hours/minutes/seconds*.
Current Cache Reset Time	The time when the statistics were last reset
Time Since Reset	The length of time since the statistics were last reset

Table 9–2 (Cont.) Oracle9iAS Web Cache Statistics

Statistic	Description
Cache Overview Table	This table provides general information about the cache: **Number of Documents in Cache**: Number of documents stored in Oracle9*i*AS Web Cache, plus the number of documents in transit through the cache. The number includes documents that have expired or have been invalidated but which have not been deleted from the cache. **Size of Documents in Cache**: The total size of the documents currently in the cache **Note:** You can adjust the maximum size of the cache in the Resource Limits page (**Cache-Specific Configuration** > **Resource Limit**). **Total Bytes Served**: Total number of bytes served to browsers **Total 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 Oracle9*i*AS Web Cache server and outgoing open connections to the origin servers. You can adjust the limit of connections in the Resource Limits page (**Cache-Specific Configuration** > **Resource Limits**). **Configured Maximum Cache Size**: The maximum cache size as specified in the Resource Limits page **Current Allocated Memory**: The physical size of the cache. The physical size of the cache is the amount of data memory allocated by Oracle9*i*AS Web Cachee for cache storage and operation. This number is always smaller than the process size shown by operating system statistics because the Oracle9*i*AS Web Cache process, like any user process, consumes memory in other ways, such as instruction storage, stack data, thread, and library data. **Current Action Limit**: Ninety percent of the **Configured Maximum Cache Size**. This number is usually larger than the **Current Allocated Memory**.

Table 9–2 (Cont.) Oracle9iAS Web Cache Statistics

Statistic	Description
Requests Served	This table provides information about the percentage of requests Oracle9*i*AS Web Cache is currently serving (**Recent** column), has served since it was started (**Since Start** column), and has served since the metrics were reset (**Since Reset** column). **Note:** If you choose **Detail Statistics**, the table shows metrics for **owned content** and **on-demand content**. These metrics are valid in a cache cluster environment. This table provides the following metrics: **Total Requests Served:** Accumulated number of browser and peer cache requests that Oracle9*i*AS Web Cache has served since it was started or restarted **Average Requests Served**: Average number of browser and peer cache requests served for each second **Fresh Hits**: Percentage of Web browser requests resolved by documents in the cache This percentage should be high, except when documents are being invalidated. **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 origin servers As documents are invalidated or expired, the percentage of stale hits will increase. The percentage will decrease as Oracle9*i*AS Web Cache retrieves updated content from the origin servers. If the percentage does not decrease, it could indicate a bottleneck on the origin servers. **Cacheable Misses**: Percentage of Web browser requests for cacheable documents not served by Oracle9*i*AS Web Cache **Noncacheable Misses**: Percentage of Web browser requests for noncacheable documents not served by Oracle9*i*AS Web Cache **Refreshes**: Percentage of documents that Oracle9*i*AS Web Cache has refreshed from the origin servers **Compressed Hits:** Percentage of total requests served out of the cache in compressed form **Compressed Misses:** Percentage of total requests retrieved from the origin server and compressed by Oracle9*i*AS Web Cache before serving

Table 9–2 (Cont.) Oracle9iAS Web Cache Statistics

Statistic	Description
Cache Errors	This table provides metrics on the apology pages served since Oracle9*i*AS Web Cache was started (**Since Start** column) or when the metrics were reset (**Since Reset** column). It lists the following metrics: **Network Errors:** Percentage of apology pages that Oracle9*i*AS Web Cache is serving to Web browsers due to a network error **Partial Page Errors:** Percentage of apology pages that Oracle9*i*AS Web Cache is serving to Web browsers due to a HTML fragment retrieval problem for a page that supports partial page caching **Site Busy Errors:** Percentage of apology pages that Oracle9*i*AS Web Cache is serving to Web browsers due to a busy Web site error
Invalidations	This table provides information metrics on the invalidation request served since Oracle9*i*AS Web Cache was started (**Since Start** column) or when the metrics were reset (**Since Reset** column). **Total Invalidation Requests:** The number of invalidation requests processed **Total Invalidation Objects:** The total number of objects invalidated

Gathering Origin Server Performance Statistics

To monitor origin server performance:

1. Start Oracle9*i*AS Web Cache Manager.

   See Also: "Starting Oracle9iAS Web Cache Manager" on page 5-3

2. In the navigator pane, select **Administration** > **Monitoring > Origin Server Statistics**.

   The Origin Server Statistics page appears.

3. From the **For Cache** list, select the cache.

   If you have not configured a cache cluster, this field displays the current cache (the cache to which you are connected.)

4. From the **Auto Refresh** list, select the frequency at which you want the statistics refreshed.

   If you select **Never**, then the page will not be refreshed again. If you want the statistics refreshed now, choose **Refresh Now**.

Table 9–3 describes the statistics for this page.

Table 9–3 Origin Server Statistics

Statistic	Description
Origin Server	This table provides information about the origin servers, that is application Web servers or proxy servers. It contains the following columns: **Origin Server**: ■ ***hostname***: Name of the origin server and the port number from which the server is listening for Oracle9*i*AS Web Cache requests. ■ ***proxy server***: **YES** specifies that the server is a proxy server. **NO** specifies that the server is an application Web server. **Up/Down Time** ■ ***up/down***: **UP** specifies that the last communication with the server was successful. **DOWN** specifies that the server is down. If this is the last server in a single or multiple server configuration, Oracle9*i*AS Web Cache keeps a connection open to the server for requests. If this is not the last server, then no new requests will be sent to server. However, other active servers will poll the downed server until it is back online. ■ ***since***: How long the origin server has been up or down **Completed Requests** ■ ***number/sec***: Number of requests that the origin server is processing for each second ■ ***max/sec***: Maximum number of requests that the origin server has processed for each second ■ ***avg/sec***: Average number of requests that the origin server has processed for each second ■ ***total***: Accumulated number of requests that the origin server has processed **Latency** ■ ***avg this interval***: Average latency for 10 second intervals to process requests for Oracle9*i*AS Web Cache ■ ***avg since start***: Average number of seconds to process requests for Oracle9*i*AS Web Cache since the origin server started

Table 9–3 (Cont.) Origin Server Statistics

Statistic	Description
	Load ■ ***now***: Current number of connections from Oracle9*i*AS Web Cache that the origin server has open ■ ***max***: Maximum number of connections that the origin server has had open at one time **Note:** Consider increasing the capacity of an origin server if the ***max*** connections is close to the server's capacity. You can increase capacity in the Application Web Servers or Proxy Servers page (**General Configuration** > **Application Web Servers** or **Proxy Servers**). **Active Sessions** ■ ***now***: Current number of active connections from Oracle9*i*AS Web Cache to the origin servers ■ ***max***: Maximum number of active connections that the origin server has had open at one time
Apology Pages Served	**# this second**: Current number apology pages that Oracle9*i*AS Web Cache is serving to Web browsers, due to a network or busy Web site error **Total**: Total number of apology pages that Oracle9*i*AS Web Cache is serving to Web browsers, due to a network or busy Web site error
Origin Server Backlog	**Now**: Current number of requests that the application Web server is processing for Oracle9*i*AS Web Cache **Max**: Maximum number of requests that the application Web server has processed for Oracle9*i*AS Web Cache

10

Troubleshooting Oracle9*i*AS Web Cache Configuration

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

This chapter contains these topics:

Startup Failures

If Oracle9*i*AS Web Cache does not start, it can be because of the following problems:

- Port Conflicts
- Startup Failure from Oracle Enterprise Manager
- 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 Oracle9*i*AS Web Cache receives browser requests. By default, the port is 7777 for HTTP requests and 4443 for HTTPS requests.

You also configure listening ports for administration, invalidation, and statistics monitoring requests. By default, the HTTP ports are 4000, 4001, and 4002, respectively. In addition to configuring listening ports for Oracle9*i*AS Web Cache, you also configure the advertised port number from which an **origin server** can receive Oracle9*i*AS Web Cache requests.

When you start Oracle9*i*AS Web Cache, a port conflict check is performed. If there is a port conflict, Oracle9*i*AS 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:

```
30/Nov/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: 7777). Change
PORT attribute of the LISTEN element in the configuration file to a suitable
unused port.
30/Nov/2001:11:04:42 -0800 -- Error: Failed to start the server.
30/Nov/2001:11:04:42 -0800 -- Error: The server could not initialize
30/Nov/2001:11:04:42 -0800 -- Information: The server is exiting
30/Nov/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 Oracle9*i*AS Web Cache Manager to resolve the port conflicts.

   Typically, Oracle9*i*AS Web Cache and the origin server ports are in conflict. Verify the port assigned to Oracle9*i*AS Web Cache in Listening Ports page (**Cache-Specific Configuration** > **Listening Ports**), and verify the host names and ports assigned to the origin servers in the Application Web Servers or Proxy Servers page (**General Configuration** > **Application Web Servers** or **Proxy Servers**).

   If there are multiple instances of Oracle9*i*AS Web Cache are running on a multihomed host with multiple IP addresses, a port conflict can occur. The IP address for the default HTTP and HTTPS ports is set to ANY. Upon startup, Oracle9*i*AS Web Cache attempts to bind the ports to all IP addresses. If a port conflict occurs, change ANY to a specific IP address in the Listening Ports page (**Cache-Specific Configuration** > **Listening Ports**).

2. Restart Oracle9*i*AS Web Cache.

See Also: "Starting and Stopping Oracle9iAS Web Cache" on page 8-2

If the administration port is in conflict, then the `admin` server process will not start and Oracle9*i*AS Web Cache Manager will not be accessible. The event log will contain messages that resemble the following output:

```
30/Nov/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.
30/Nov/2001:10:56:11 -0800 -- Error: Failed to start the server.
30/Nov/2001:10:56:11 -0800 -- Error: The server could not initialize
30/Nov/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"/>
  <MULTIPORT>
    <LISTEN IPADDR="ANY" PORT="1100" PORTTYPE="NORM"/>
    <LISTEN IPADDR="ANY" PORT="4000" PORTTYPE="ADMINISTRATION"/>
    <LISTEN IPADDR="ANY" PORT="4003" PORTTYPE="INVALIDATION"/>
    <LISTEN IPADDR="ANY" PORT="4002" PORTTYPE="STATISTICS"/>
  </MULTIPORT>
```

Startup Failure from Oracle Enterprise Manager

If you do not configure Oracle9*i*AS Web Cache during installation and attempt to start it at a later time from **Oracle Enterprise Manager**, then Oracle Enterprise Manager may return the following error when there is a port conflict with another running Oracle9*i*AS Web Cache in a different Oracle home on the same computer.

```
oracle.sysman.emSDK.util.jdk.EMException: Throwable: The attempt to start or
stop Web Cache failed.
```

To resolve this issue, perform the following steps to the Oracle9*i*AS Web Cache that was not configured during installation:

1. Modify the Oracle9*i*AS Web Cache listening ports in the Listening Ports page (**Cache-Specific Configuration** > **Listening Ports**) so they are not in conflict with the other Oracle9*i*AS Web Cache.

   See Also: "Task 3: Configure Oracle9iAS Web Cache with Listening Ports for Incoming Browser Requests" on page 6-6

2. Change the Oracle9*i*AS Web Cache listening port value in the `targets.xml` file located in `$ORACLE_HOME/sysman/emd/targets.xml` on UNIX or *`ORACLE_HOME`*`\sysman\emd\targets.xml` on Windows.
3. Modify the `PORT` directive in `httpd.conf` to the Oracle9*i*AS Web Cache port number.

   See Also: "Task 4: Provide Directives to Oracle HTTP Server" on page 6-8

Cache Memory

Oracle9*i*AS Web Cache preallocates a large memory pool for data storage. When Oracle9*i*AS 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` file:

```
30/Nov/2001:10:58:02 -0600 -- Error: Oracle Web Cache Cache failed to initialize
30/Nov/2001:10:58:02 -0600 -- Error: The server could not initialize
30/Nov/2001:10:58:02 -0600 -- Information: The server is exiting
30/Nov/2001:10:58:02 -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 Oracle9*i*AS Web Cache.

See Also: "Starting and Stopping Oracle9iAS Web Cache" on page 8-2

Privileged Ports

Port numbers less than 1024 are reserved for use by privileged processes on UNIX. If you want to configure Oracle9*i*AS Web Cache to listen on a port less than 1024, such as on port 80, then the `webcachectl` executable needs to run as root. In order to run as the root user, ensure that `root.sh` was run during installation. If `root.sh` was not run during installation, run it now from the `$ORACLE_HOME` directory.

If `webcachectl` cannot run as root, then the `cache` server process fails to start and messages that resemble the following output are written to the `event_log` file:

```
30/Nov/2001:23:06:51 +0000 -- Error: Unable to listen on port 80
30/Nov/2001:23:06:51 +0000 -- Error: Failed to start the server.
30/Nov/2001:23:06:51 +0000 -- Error: The server could not initialize
30/Nov/2001:23:06:51 +0000 -- Information: The server is exiting
```

Greater Than One Thousand Maximum Connections

If you need Oracle9*i*AS Web Cache to support more than 1,024 connections on UNIX, then the `webcachectl` executable needs to run as root. In order to run as the root user, ensure that `root.sh` was run during installation. If `root.sh` was not run during installation, run it now from the `$ORACLE_HOME` directory.

If `webcachectl` cannot run as root, then the `cache` server process fails to start and messages that resemble the following output are written to the `event_log` file:

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

See Also: "Connection Limit" on page 6-19

Wallet Cannot Be Opened

When Oracle9*i*AS Web Cache is unable to open a **wallet**, messages that resemble the following output are written to the `event_log` file:

```
30/Nov/2001:23:13:41 +0000 -- Information: SSLInitialize: Origin Server Wallet
file /etc/ORACLE/WALLETS/janedoe/ewallet.p12 does not exist
30/Nov/2001:23:13:41 +0000 -- Information: SSLInitialize: Origin Server Wallet
Autologin file /etc/ORACLE/WALLETS/janedoe/cwallet.sso
does not exist - Wallet does not appear to be autologin wallet
30/Nov/2001:23:13:41 +0000 -- Warning: Origin Server Wallet Failed to open at
location /etc/ORACLE/WALLETS/janedoe NZERROR=28759
-- Opened wallet as user=janedoe. -- Please verify wallet location and Auto
Login support enabled.
30/Nov/2001:23:13:41 +0000 -- Warning: SSLInitialize: Orgin Server Wallet did
not open -- Operating without wallet for backend -- Only
Diffie-Hellman Anonymous Connections Supported to Origin Servers!
30/Nov/2001:23:13:41 +0000 -- Information: SSLInitialize: Wallet file
/etc/ORACLE/WALLETS/janedoe/ewallet.p12 does not exist
30/Nov/2001:23:13:41 +0000 -- Information: SSLInitialize: Wallet Autologin file
/etc/ORACLE/WALLETS/janedoe/cwallet.sso does not exist - Wallet does not appear
to be autologin wallet
30/Nov/2001:23:13:41 +0000 -- Warning: Wallet Failed to open at location
/etc/ORACLE/WALLETS/janedoe NZERROR=28759 -- Opened wallet as user=janedoe. --
Please verify wallet location and Auto Login support enabled.
30/Nov/2001:23:13:41 +0000 -- Error: The server could not initialize
30/Nov/2001:23:13:41 +0000 -- Information: The server is exiting
```

To resolve this error, perform the procedure that follows. At the end of each step, restart Oracle9*i*AS Web Cache with the `webcachectl restart` command, and recheck the `event_log` file for wallet errors.

1. Follow the "Enabling Wallets to Open on Windows" on page 6-39 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
3. Ensure that wallet files `cwallet.sso` and `ewallet.der` exist.

   If these files do not exist, then an auto-login wallet does not exist. In this case, create the wallet.

   See Also: "Task 1: Create Wallets" on page 6-39

   If these files do exist, then the wallet user may not match the Oracle9*i*AS Web Cache user. Continue to Step 4.
4. Change the owner of the wallet to the user ID and group ID specified in the Process Identity page (**Cache-Specific Configuration** > **Process Identity**) of the Oracle9*i*AS Web Cache Manager.

   See Also: "Task 2: Modify Security Settings" on page 6-3
5. Confirm that the wallet was not copied from one computer to another.

   See Also:

   - "Task 2: Modify Security Settings" on page 6-3 for instructions on setting the user ID of the Oracle9*i*AS Web Cache executables
   - "Task 1: Create Wallets" on page 6-39 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, re-create the wallet.

   See Also: "Task 1: Create Wallets" on page 6-39

Caching Rules

To diagnose if caching rules are serving wrong or older content:

1. Analyze the content of the `event_log` file and the `access_log` file in `$ORACLE_HOME/webcache/logs` on UNIX and in *`ORACLE_HOME`*`\webcache\logs` on Windows.

   If verbose logging is turned on in the event log, then error messages about the caching rules are reported.

   See Also: "Configuring Event Logs" on page 8-56 for further information about turning on verbose logging

2. Determine the contents of the cache by:

   - Listing the most popular objects in the cache, along with the caching rules associated with the documents
   - Listing the contents of the cache
   - Previewing invalidation without invalidating actual content

   See Also:

   - "Listing Popular Documents" on page 8-49 to generate a list of the URLs of the most popular documents stored in the cache since the cache was last started
   - "Listing All Contents" on page 8-50 to generate a list of the URLs of the objects currently in the cache
   - Step 5 in "Manual Invalidation Using Oracle9iAS Web Cache Manager" on page 8-24 or "Invalidation Preview Request Syntax" on page 8-17 to preview invalidation

3. Compare the contents of the cache to the caching rules in the Cacheability Rules page (**General Configuration** > **Cacheability Rules**) in Oracle9*i*AS Web Cache to determine discrepancies.
4. Adjust caching rules by adding or removing rules, adjusting regular expression syntax, or changing the precedence of rules.

   See Also: Chapter 7, "Creating Caching Rules"

Load on Oracle9*i*AS Web Cache Computer

On UNIX operating systems, the `top` and `uptime` utilities report a higher than expected average load when the Oracle9*i*AS Web Cache computer is idle. This occurs because Oracle9*i*AS 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. Oracle9*i*AS Web Cache increases the average queue length (uptime output) by approximately one.
- The CPU load is still low because the work Oracle9*i*AS Web Cache performs is minimal.

Diagnostic Information in the Server Response-Header Field or HTML Body

By default, Oracle9*i*AS Web Cache adds diagnostics information to the `Server`-response header field:

`Server: Oracle9iAS/`*`version server_header_from_origin_server`*
`Oracle9iAS-Web-Cache/`*`version`* `(`*`diagnostic_information`*`)`

For diagnostics purposes, it can be useful to also display this information in the HTML response body of a document.

> **See Also:** "Server Response-Header Field" on page 2-40 for an overview of the `Server` response-header field

To configure diagnostic information to display in the HTML response body:

1. Use a text editor to open the `webcache.xml` file.
2. Locate the `DEBUGINFO HEADER` attribute in the `SECURITY` element:

```
<?xml version="1.0"?>
<!DOCTYPE CALYPSO SYSTEM "internal:///webcache.dtd">
<CALYPSO>
<VERSION DTD_VERSION="2.0.4"/>
...
  <SECURITY>
    <USER .../>
    <USER .../>
    <SECURESUBNET ...>
    </SECURESUBNET>
```

```
    <DEBUGINFO HEADER="YES" BODY="NO"/>
  </SECURITY>
...
```

3. Change the value of the `BODY` attribute from `NO` to `YES`.
4. Restart Oracle9*i*AS Web Cache with the `webcachectl restart` command.
5. Append the string `+wcdebug` to the URL of the document into the browser to see the diagnostic information string embedded in the response body.

Note the following limitations when appending the string `+wcdebug` to the URL of the document to see the diagnostic information string embedded in the response body:

- In a cache cluster, if a cache member receives a request with `+wcdebug` for a content owned by another cache member, the page with debug information may be stored in the cache as an on-demand cached page. During a subsequent request for the same page without `+wcdebug`, you will retrieve the on-demand cached page with the debug information.
- Appending debug information to compressed pages causes an error in the browser. If you access the page with `telnet`, you will see the debug information prepended to the compressed page.

If you do not want users viewing the Oracle9*i*AS Web Cache diagnostic information in the `Server`-response header field, modify the `webcache.xml` file to disable it.

To disable the display of the diagnostic information:

1. Use a text editor to open the `webcache.xml` file.
2. Locate the `DEBUGINFO HEADER` attribute in the `SECURITY` element:

```
<?xml version="1.0"?>
<!DOCTYPE CALYPSO SYSTEM "internal:///webcache.dtd">
<CALYPSO>
<VERSION DTD_VERSION="2.0.4"/>
...
  <SECURITY>
    <USER .../>
    <USER .../>
    <SECURESUBNET ...>
    </SECURESUBNET>
    <DEBUGINFO HEADER="YES" BODY="NO"/>
  </SECURITY>
...
```

3. Change the value of the HEADER attribute from YES to NO.
4. Ensure that the value of the BODY attribute is NO.
5. Restart Oracle9*i*AS Web Cache with the `webcachectl restart` command.

Invalidation Time-outs

The **invalidation** feature has a default time-out of 300 seconds for the propagation of invalidation requests in the **cache hierarchy** or **cache cluster** deployments.

> **See Also:** "Invalidation Propagation" on page 2-7 for an overview of invalidation propagation

When the time-out is exceeded in a cache hierarchy, a message that resembles the following output is written to the `event_log` file of the **remote cache** or **subscriber cache**:

```
22/Apr/2002:18:34:55 -0700 -- Information: <Invalidation>Invalidation sent
upstream to webcache host '130.35.45.41' port '22002' has returned with response
code: 'failed-no response code'.
```

To resolve this error:

1. On the central or provider cache, use a text editor to open the `webcache.xml` file.
2. Locate the CALYPSONETINFO element:

   ```
   <CALYPSONETINFO...INV_PEER_TIMEOUT="300"
                    INV_GLOBAL_TIMEOUT="300".../>
   ```

3. Modify the value of the INV_GLOBAL_TIMEOUT attribute to a larger value.

   The higher the value, the more system resources that will be used. If the network is fast, only increase the value to what is needed.
4. Restart Oracle9*i*AS Web Cache with the `webcachectl restart` command.

When the time-out is exceeded in a cache cluster, a message that resembles the following output is displayed in the Cache Cleanup Results dialog box or in the response to the invalidation request:

```
Can't connect to the web cache's invalidation listening port.
```

1. On cache cluster members, use a text editor to open the `webcache.xml` file.
2. Locate the `CALYPSONETINFO` element:

   ```
   <CALYPSONETINFO...INV_PEER_TIMEOUT="300"
                    INV_GLOBAL_TIMEOUT="300".../>
   ```

3. Modify the value of `INV_PEER_TIMEOUT` attribute.

   In a cache cluster, it is likely that cache cluster members are running in a LAN environment. Therefore, decreasing the value of `INV_PEER_TIMEOUT` will typically improve efficiency.

4. Restart Oracle9*i*AS Web Cache with the `webcachectl restart` command.

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 Resource Limits page (**Cache-Specific Configuration** > **Resource Limits**) of Oracle9*i*AS Web Cache Manager, check the value of the **Maximum Incoming Connections** field. This field provides the currently configured capacity. If the capacity can be adjusted, increase it.

  See Also: "Task 9: Specify Settings for Origin Servers" on page 6-23

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

  See Also: Chapter 7, "Creating Caching Rules"

Content-Length Request-Header Field

If the application Web server sets the `Content-Length` request-header field to some number of bytes, then Oracle9*i*AS Web Cache, which is configured by default for a five second connection time-out, keeps the connection open until the browser has received the bytes or the connection times out.

If the actual length of the page is less than the `Content-Length`, then the browser expects more data to arrive and the connection will eventually time out. If the actual length of the page is greater than the `Content-Length`, then the browser will not receive the complete page. This problem does not occur for cache hits because Oracle9*i*AS Web Cache correctly calculates the `Content-Length` itself when inserting pages into the cache repository. For cache misses, there are two workarounds for the improper `Content-Length` problem:

- Fix your application to ensure that `Content-Length` is correctly reported
- Configure the browser or client emulator to send HTTP/1.0 requests without the `Keep-Alive` request-header field

Because many servers are not configured to send the `Keep-Alive` request-header field, they disconnect after sending the last byte of data. Therefore, you may not see this issue when querying the application Web server directly from a client emulation tool or browser. Without the `Keep-Alive` request-header field, most browsers disregard the discrepancies between `Content-Length` and the actual data received.

HTTP 500 Response Status Codes

Oracle9*i*AS Web Cache does not cache pages that generate HTTP 500 response status codes. However, if an application reports an exception and does not send the error code, then the exception may be cached by Oracle9*i*AS Web Cache.

When you are developing a servlet and JSP and want to cache the content of the servlet and JSP, ensure that you send an non-200 HTTP error code in the runtime exception handling code. Otherwise, if the document has a cacheability rule associated with it, then Oracle9*i*AS Web Cache caches the document with the exception until the document is invalidated.

If you are using OracleJSP on Jserv as your JSP engine and servlet engine, then OracleJSP will generate HTTP 200 status code exceptions when a JSP file is not found or there is a compilation error in a JSP file. This is mainly for ease of debugging of JSP development in a JServ environment.

To enable OracleJSP on Jserv to generate an HTTP `404 Not Found` or 500 response status codes rather than HTTP 200 status codes, set `send_error` to `true` and `unsafe_reload` to `true` in `zone.properties`. `zone` in your Jserv deployment environment.`zone.roperties` is located in `$ORACLE_HOME/Apache/Jserv/etc` on UNIX or *`ORACLE_HOME`*`\Apache\Jserv\etc` on Windows.

When using OracleJSP on OC4J, it is not necessary to establish these settings. OC4J always sends 500 status codes when it encounters a compilation error.

See Also:

- `http://www.ietf.org/rfc/rfc2616.txt` for further information about HTTP status codes
- *Oracle HTTP Server Administration Guide* for further information about Jserv

Administrator Password in the Change Administration Password Dialog Box

If you change the `administrator` password in the Change Administration Password dialog box (**General Configuration** > **Security**) without using the `webcachectl restart` command, and attempt to re-change the password again, the following error appears:

```
Incorrect old password.
```

The resolve this problem, remove the `.webcache_tmp*.*` file in `$ORACLE_HOME/webcache` on UNIX and *`ORACLE_HOME`*`\webcache` on Windows.

Changes to the `administrator` password made in the Change Administration Password dialog box are saved initially to a temporary version of the `webcache.xml` file named `.webcache_tmp*.*`. This file is stored in `$ORACLE_HOME/webcache` on UNIX and *`ORACLE_HOME`*`\webcache` on Windows. Until the new password is committed to the `webcache.xml` file with the `webcachectl restart` command, Oracle9*i*AS Web Cache Manager uses the configuration changes that were in process by using the temporary file. As a result, the Change Administration Password dialog uses the password in the temporary file for the current password.

Browser-Specific Issues

Table 10–1 describes browser limitations and their impact on Oracle9*i*AS Web Cache.

Table 10–1 Browser Issues

Problem	Description
Compressing JavaScript Files	**Problem Description:** Compressed JavaScript files cause some Netscape browsers to behave erratically and possibly fail. This issue only effects files that are referenced with the `src` attribute of the `script` tag; it does not include files that contain inline JavaScripts. **Known browsers effected:** Netscape 4.*x* **Example:** `<script language="JavaScript" src="copyright.js"></script>` If `copyright.js` is compressed for Netscape, then the browser may fail. **Workaround:** By default, compression is turned off for included JavaScript files. View the Cacheability Rules page to see this setting (**General Configuration > Cacheability Rules**).

Compressing Documents with `Content-Disposition` Response-Header Fields	**Problem Description:** Documents with `Content-Disposition` response-header fields show incorrect file names when they are compressed.
	
	Known browsers effected: Internet Explorer 5.0, 5.5, and 6.0
	
	Example: Response headers for URL `/reportgen` include to following:
	
	`Content-Type: application/excel`
	
	`Content-Disposition: attachment; filename="file.csv"`
	
	When the document is not compressed, a **Save As** dialog appears with `file.csv` as the default filename. However, if it is compressed, `reportgen` appears as the default name. Without the correct extension, the file will not open correctly on Windows operating systems.
	
	Workaround: Even if compression is selected, Oracle9*i*AS Web Cache does not compress documents containing a `Content-Disposition` response-header field.

Table 10–1 (Cont.) Browser Issues

Problem	Description
Decompressing Documents with `Content-Disposition` Response-Header Fields	**Description:** Documents with `Content-Disposition` response-header fields are not decompressed when you choose **File** > **Save As** from Netscape browsers. **Known browsers effected:** Netscape 4.*x* **Workaround:** Even if compression is selected, Oracle9*i*AS Web Cache does not compress documents containing a `Content-Disposition` response-header field.
Compressing Style Sheets	**Description:** Compressed style sheets can cause background attributes, such as background images, to not appear in the output. **Known browsers effected:** Internet Explorer 6.0 **Workaround:** Disable compression for style sheets in the Cacheability Rules page (**General Configuration > Cacheability Rules**).
Keep-Alive	**Problem Description:** For HTTPS requests, Internet Explorer 5.5 may send requests after Oracle9*i*AS Web Cache has already tried to close the connection. **Known browsers effected:** Internet Explorer 5.5 **Workaround:** Disable keep-alive by setting Keep-Alive to 0 in the Network Timeouts page (**Cache-Specific Configuration** > **Network Timeouts**) of the Oracle9*i*AS Web Cache Manager. This closes the connection between Oracle9*i*AS Web Cache and the browser after the browser returns a response.

Part III

Reference

Part III provides reference material for this guide.

This part contains these appendixes:

- Appendix A, "Oracle9iAS Web Cache Directory Structure"
- Appendix B, "Oracle9iAS Web Cache Default Settings"
- Appendix C, "Invalidation and Statistics Document Type Definitions"
- Appendix D, "Edge Side Includes Language"
- Appendix E, "Event Log Messages"
- Appendix F, "Using Oracle9iAS Web Cache with Third-Party Application Web Servers"

A

Oracle9*i*AS Web Cache Directory Structure

This appendix describes the installed Oracle9*i*AS Web Cache directory structure.

When you install Oracle9*i*AS Web Cache, all subdirectories are under a top-level directory of `$ORACLE_HOME/webcache` on UNIX and *`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 Oracle9iAS Web Cache Directory Structure

Directory/File	Contents
/bin directory on UNIX	Contains the Oracle9*i*AS Web Cache the following executables: ■ webcached runs Oracle9*i*AS Web Cache ■ webcachemon executable manages the auto-restart process ■ webcachectl executable on UNIX enables you to manage the **admin server process**, **cache server process**, and **auto-restart process** **Note:** On Windows, executables are located in the *ORACLE_HOME**bin* directory.
/docs directory on UNIX \docs directory on Windows	Contains documentation and online help for Oracle9*i*AS Web Cache Manager. It also contains apology pages.
/dtds directory on UNIX \dtds directory on Windows	Contains the following Document Type Definition (DTD) files: ■ internal.dtd for the internal.xml file ■ wcstats.dtd for statistic monitoring requests and responses ■ webcache.dtd for the webcache.xml file ■ webcache20.dtd for the webcachemigtool utility during migration
/examples directory on UNIX \examples directory on Windows	Contains source code and scripts that enable you to better customize applications for Oracle9*i*AS Web Cache features **See Also:** readme.examples.html in the directory for further information
/jlib directory on UNIX \jlib directory on Windows	Contains a Java invalidation API **See Also:** *Invalidation API Reference (Javadoc)*
/lib directory on UNIX	Contains library files

Table A–1 (Cont.) Oracle9iAS Web Cache Directory Structure

Directory/File	Contents
`/invalidation` directory on UNIX `\invalidation` directory on Windows	Contains the Document Type Definition (DTD) file `WCSinvalidation.dtd` for invalidation requests and responses. It also contains the `invalidation.dtd` file for invalidation requests and responses in the release 1.0 format.
`/logs` directory on UNIX `\logs` directory on Windows	Contains event and access logs
`/mesg` directory on UNIX `\mesg` directory on Windows	Contains message files
`internal.xml` and `internal_admin.xml` files	Contains internal configuration settings
`webcache.pid` file on UNIX	Contains the process ID of the `cache` server process. The `admin` server process, `auto-restart` process, and the `webcachectl` utility read this file when working with the `cache` server process.
`webcache.xml` file	Contains configuration parameters set by Oracle9*i*AS Web Cache Manager
`webcacheadmin.pid` file on UNIX	Contains the process ID of the `admin` server process
`webcachemon.pid` file on UNIX	Contains the process ID of the `auto-restart` monitor process
`webcachetargets.xml` file	Contains configuration parameters used by Oracle Enterprise Manager

B

Oracle9*i*AS Web Cache Default Settings

Oracle9*i*AS 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 Oracle9*i*AS Web Cache Manager interface you can change the values.

Table B–1 Oracle9iAS Web Cache Default Settings

Configuration Settings	Default Value	Location in Oracle9*i*AS Web Cache Manager to Change Value
Security		
Password for the `administrator` user	`administrator`	**General Configuration** > **Security**
Password for the `invalidator` user	`invalidator`	**General Configuration** > **Security**
Process identify for Oracle9*i*AS Web Cache	User and group ID of user that installed Oracle9*i*AS Web Cache	**Cache-Specific Configuration** > **Process Identity**
Ports		
Oracle9*i*AS Web Cache	HTTP: 7777 HTTPS: 4443	**Cache-Specific Configuration** > **Listening Ports**
Administration	4000	**Cache-Specific Configuration** > **Operations Ports**
Invalidation	4001	**Cache-Specific Configuration** > **Operations Ports**
Statistics	4002	**Cache-Specific Configuration** > **Operations Ports**

Table B–1 (Cont.) Oracle9iAS Web Cache Default Settings

Configuration Settings	Default Value	Location in Oracle9*i*AS Web Cache Manager to Change Value
Network Time-outs		
Keep-Alive time-outs	5 seconds	**Cache-Specific Configuration** > **Network Timeouts**
Origin server time-out	300 seconds	**Cache-Specific Configuration** > **Network Timeouts**
Resource Limits		
Maximum cache size	500 MB	**Cache-Specific Configuration** > **Resource Limits**
Maximum incoming connections	900	**Cache-Specific Configuration** > **Resource Limits**
Origin Server Failover		
Failover threshold	5	**General Configuration** > **Application Web Servers** or **Proxy Servers**
Polling interval for a failed origin server	10 seconds	**General Configuration** > **Application Web Servers** or **Proxy Servers**
Apology Pages		
Network error	`network_error.html` in `$ORACLE_HOME/webcache/docs` on UNIX and *`ORACLE_HOME`*`\webcache\docs` on Windows	**General Configuration** > **Apology Pages**
Site busy error	`busy_error.html` in `$ORACLE_HOME/webcache/docs` on UNIX and *`ORACLE_HOME`*`\webcache\docs` on Windows	**General Configuration** > **Apology Pages**
Partial page error	None	**General Configuration** > **Apology Pages**

Table B–1 (Cont.) Oracle9iAS Web Cache Default Settings

Configuration Settings	Default Value	Location in Oracle9*i*AS Web Cache Manager to Change Value
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	**Cache-Specific Configuration** > **Access Logging** to modify the default directory location **Note:** The file name cannot be modified.

C

Invalidation and Statistics Document Type Definitions

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

This appendix contains these topics:

Invalidation DTD

This section describes the DTD of invalidation requests and responses. The DTD for both requests and responses is defined within `WCSinvalidation.dtd`, located in the `$ORACLE_HOME/webcache/invalidation` directory on UNIX and the *`ORACLE_HOME`*`\webcache\invalidation` directory on Windows.

This section contains the following topics:

Invalidation Request and Response DTD

Figure C–1 shows the portion of the DTD for invalidation requests.

Figure C–1 Invalidation Request DTD

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

<!-- VERSION is currently "WCS-1.1" without the quotes -->
<!ATTLIST    INVALIDATION
             VERSION          CDATA          #REQUIRED
>

<!ELEMENT    SYSTEM           (SYSTEMINFO+)>

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

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

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

<!ELEMENT    ADVANCEDSELECTOR (COOKIE|HEADER|OTHER)*>
<!ATTLIST    ADVANCEDSELECTOR
             URIPREFIX        CDATA          #REQUIRED
             HOST             CDATA          #IMPLIED
             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    OTHER           EMPTY>
<!ATTLIST    OTHER
             TYPE            CDATA          #REQUIRED
             NAME            CDATA          #REQUIRED
             VALUE           CDATA          #IMPLIED
>

<!ELEMENT    ACTION          EMPTY>
<!ATTLIST    ACTION
             REMOVALTTL      CDATA          #IMPLIED
>
<!ELEMENT    INFO          EMPTY>
<!ATTLIST    INFO          VALUE       CDATA          #REQUIRED
>
```

Table C–1 Invalidation Request DTD Elements and Attributes

Element	Attribute	Description
INVALIDATION		Root element
	VERSION	Version of the WCSinvalidation.dtd file to use as the XML document type
SYSTEM		
SYSTEMINFO		The possible NAME/VALUE pair in a request is as follows: NAME="WCS_PROPAGATE" VALUE="TRUE|FALSE" This pair specifies whether or not invalidation requests are propagated to cache cluster members. If WCS_PROPAGATE is TRUE, it overrides the setting for invalidation propagation in the configuration. If WCS_PROPAGATE is FALSE, it uses the setting specified in the configuration.

Table C–1 (Cont.) Invalidation Request DTD Elements and Attributes

Element	Attribute	Description
OBJECT		
	BASICSELECTOR	Invalidation based on the URL
	ADVANCEDSELECTOR	Invalidation based on advanced invalidation selectors
	ACTION	Action to perform on documents
BASICSELECTOR		
	URI	URL of the documents to be invalidated
ADVANCEDSELECTOR		
	URIPREFIX	Prefix path of the documents to be invalidated The prefix path must begin with `http|https://host_name:port/path/filename` or with "/" and end with "/". `http|https://host_name:port/path/filename` is required if the HOST attribute is not specified. The prefix is interpreted literally, including reserved **regular expression** characters.
	URIEXP	URL of the documents to be invalidated underneath the URIPREFIX
	HOST	Host name and port number of the site (*host_name*:*port*) Port 80 is the default port for HTTP and port 443 is the default port for HTTPS.
	METHOD	**HTTP request method** of the documents to be invalidated
	BODYEXP	HTTP POST body message of the documents 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.

Table C–1 (Cont.) Invalidation Request DTD Elements and Attributes

Element	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.
OTHER		
	NAME	NAME is one of the following: ■ URI to specify a match of the URL specified in VALUE ■ BODY to specify a match of the HTTP POST body ■ QUERYSTRING_PARAMETER to specify a match of an embedded URL parameter
	TYPE	TYPE is one of the following: ■ SUBSTRING to specify a substring match ■ REGEX to specify a regular expression match
	VALUE	Value of URI, BODY, or QUERYSTRING_PARAMETER
ACTION		
	REMOVALTTL	Maximum time that documents can reside in the cache before they are invalidated. The default is 0 seconds.
INFO		
	VALUE	Comment for the documents to be included in the invalidation result

Figure C–2 shows the portion of the DTD for invalidation responses.

Figure C–2 Invalidation Response DTD

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

<!-- VERSION is currently "WCS-1.1" without the quotes -->
<!ATTLIST    INVALIDATIONRESULT
             VERSION          CDATA          #IMPLIED
>

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

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

Table C–2 Invalidation Response DTD Elements and Attributes

Element	Attribute	Description
INVALIDATIONRESULT		
	VERSION	Version of the WCSinvalidation.dtd file to use as the XML document type
SYSTEM		
SYSTEMINFO		The possible NAME/VALUE pair in a response is as follows: NAME="WCS_CACHE_NAME" VALUE="*string*" This pair specifies the name of the cache.
OBJECTRESULT		
	BASICSELECTOR	Invalidation based on the URL
	ADVANCEDSELECTOR	Invalidation based on advanced invalidation selectors
	RESULT	Action to perform on documents
	INFO	Comment specified in the INFO element of the invalidation request

Table C–2 (Cont.) Invalidation Response DTD Elements and Attributes

Element	Attribute	Description
BASICSELECTOR		**See Also:** "BASICSELECTOR" on page C-5
ADVANCEDSELECTOR		**See Also:** "ADVANCEDSELECTOR" on page C-5
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 is one of the following: ■ 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

Invalidation Preview Request and Response DTD

Figure C–3 shows the portion of the DTD for invalidation preview requests.

Figure C–3 Invalidation Request DTD

```
<!-- root element for invalidation preview request -->
<!ELEMENT INVALIDATIONPREVIEW (SYSTEM?, (BASICSELECTOR|ADVANCEDSELECTOR))>

<!-- VERSION is currently "WCS-1.1" without the quotes -->
<!ATTLIST INVALIDATIONPREVIEW
          VERSION  CDATA #REQUIRED
          STARTNUM CDATA #REQUIRED
          MAXNUM   CDATA #REQUIRED
>
```

Table C–3 Invalidation Preview Request DTD Elements and Attributes

Element	Attribute	Description
INVALIDATIONPREVIEW		
	VERSION	Version of the WCSinvalidation.dtd file to use as the XML document type
	STARTNUM	Number representing the first document to be listed
	MAXNUM	Number of documents to be listed
SYSTEM		
SYSTEMINFO		The possible NAME/VALUE pair in a request is as follows: NAME="WCS_PROPAGATE" VALUE="TRUE|FALSE" This pair specifies whether or not invalidation requests are propagated to cache cluster members. If WCS_PROPAGATE is TRUE, it overrides the setting for invalidation propagation in the configuration. If WCS_PROPAGATE is FALSE, it uses the setting specified in the configuration.
BASICSELECTOR		**See Also:** "BASICSELECTOR" on page C-5
ADVANCEDSELECTOR		**See Also:** "ADVANCEDSELECTOR" on page C-5

Figure C–4 shows the portion of the DTD for invalidation preview responses.

Figure C–4 Invalidation Preview Response DTD

```
<!-- root element for invalidation preview result -->
<!ELEMENT INVALIDATIONPREVIEWRESULT (SYSTEM?, SELECTEDURL*)>

<!ATTLIST INVALIDATIONPREVIEWRESULT
          VERSION      CDATA #REQUIRED
          STATUS       CDATA #REQUIRED
          STARTNUM     CDATA #REQUIRED
          NUMURLS      CDATA #REQUIRED
          TOTALNUMURLS CDATA #REQUIRED
>

<!ELEMENT SELECTEDURL EMPTY>
<!ATTLIST SELECTEDURL VALUE CDATA #REQUIRED>
```

Table C–4 Invalidation Preview Response DTD Elements and Attributes

Element	Attribute	Description
INVALIDATIONPREVIEWRESULT		
	VERSION	Version of the WCSinvalidation.dtd file to use as the XML document type
	STATUS	Status of the preview. Status is one of the following: ■ SUCCESS for successful invalidations ■ URI NOT CACHEABLE for documents that are not cacheable ■ URI NOT FOUND for documents not found
	STARTNUM	Number representing the first document to be listed
	NUMURLS	Number of URLs returned in this preview result
	TOTALNUMURLS	Number of URLs matching the BASICSELECTOR or ADVANCEDSELECTOR selectors
SELECTEDURL		URLs matching the BASICSELECTOR or ADVANCEDSELECTOR selectors to be invalidated
BASICSELECTOR		**See Also:** "BASICSELECTOR" on page C-5
ADVANCEDSELECTOR		**See Also:** "ADVANCEDSELECTOR" on page C-5

Statistics DTD

Oracle9*i*AS Web Cache provides a simple, extensible, and flexible XML query interface to monitor the cache runtime information. You can query any selected group of data to get information about the performance of the cache, as well as information about the cache configuration.

This section describes the DTD of statistics requests and responses. The DTD for both requests and responses is defined within the file `wcstats.dtd`, located in the `$ORACLE_HOME/webcache/dtds` directory on UNIX and the *`ORACLE_HOME`*`\webcache\dtds` directory on Windows.

Figure C–5 shows the contents of the statistics DTD file, `wcstats.dtd`.

Figure C–5 Statistics DTD

```
<?xml version="1.0"?>
<!ELEMENT WCSTATS (GROUP*)>

<!ATTLIST WCSTATS DTD_VERSION CDATA #FIXED "1.0">

<!ELEMENT GROUP (PARAM*, ENTRY*, GROUP*)>
<!ATTLIST GROUP
  NAME CDATA #REQUIRED
>

<!ELEMENT PARAM EMPTY>
<!ATTLIST PARAM
  NAME CDATA #REQUIRED
  VALUE CDATA #REQUIRED
>

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

Table C–5 describes the element names, the attributes, and descriptions of each attribute.

Table C–5 Statistics DTD

Element	Attribute	Description
WCSTATS		Root element
	DTD_VERSION	The version of the DTD to use as the XML document type. The valid value is 1.0.
	GROUP	You can specify more than one GROUP element in the request. GROUP can have child elements, including subgroups. If a request contains the GROUP element without any children, Oracle9*i*AS Web Cache returns all of the ENTRY values for the group as well as all subgroup values.
	NAME	A unique string associated with each statistic or group of statistics. In each top-level GROUP, NAME is associated with a unique numerical ID. You can use either the string or the numerical ID.
PARAM		An optional subelement of the element GROUP. This element is used to send input parameters. PARAM cannot have child elements. For example, to request URL_STATISTICS, you specify how many URLs are to be returned in the result. You pass the string "OBJECT_COUNT" as the NAME and the number of URLs to be returned as the VALUE.
	NAME	A string that names the data to be returned
	VALUE	A string used to pass values to the element
ENTRY		An optional subelement of the element GROUP. ENTRY cannot have child elements.
	NAME	A unique string associated with each statistic
	VALUE	A string used to pass values to the element or to retrieve values from Oracle9*i*AS Web Cache

Groups of Statistics

In the statistics DTD, types of data are grouped based on their logical relationship. The statistics DTD associates a group ID with each top-level group. To improve the performance of group lookup, you can use the number in the GROUP NAME field in your query XML message, rather than the string. (String comparison is a slower operation.)

To ensure the backward compatibility of all the XML messages, future versions of Oracle9*i*AS Web Cache will add new groups only after existing groups, without changing the existing order. If, in future versions, Oracle9*i*AS Web Cache deletes any existing groups, that group ID will not be reused for other purposes.

Table C–6 shows the group names and their corresponding group IDs.

Table C–6 Statistics Groups

Group Name	Group ID
TIME	101
PID	102
OPEN_CONNECTIONS	103
BYTES_SERVED	104
BYTES_SAVED_WITH_COMPRESSION	105
INVALIDATION_REQUESTS	106
INVALIDATED_OBJECTS	107
CACHED_DOC_COUNT	108
CACHED_DOC_SIZE	109
REFRESHES	114
COMPRESSED_HITS	115
COMPRESSED_MISSES	116
SESSION_COUNT	121
URL_STATS	124
CACHEABILITY_RULES	125
HTTP_REQUESTS	126
ERRORS	127

Table C–6 (Cont.) Statistics Groups

Group Name	Group ID
HITS	128
MISSES	129
CLUSTERS	130
SITE	131
HTTP_CLIENT_REQUESTS	132
CACHE_INFO	133

These groups are divided into five main categories:

- Cache Information Groups
- Runtime Statistics Groups
- Site Information Group
- Origin Server Statistics Group
- URL Statistics Group

Cache Information Groups

The cache information groups provide general information about caches.

Table C–7 lists the cache information groups and subgroups, the valid values that can be passed to the NAME attribute of the ENTRY element for the group, and a description of the attribute.

Table C–7 Cache Information Group

GROUP Name	ENTRY Name	Description
TIME		A group that returns information about how long the cache has been running and how long since the statistics were reset
	CACHE_START_TIME	The time when the cache was last started
	STATS_RESET_TIME	The time when the statistics were last reset
	LAST_MODIFIED_ TIME	The time when the cache was last modified
PID		A group that returns information about the cache server ID
	CACHE_PROCESS	The process ID of the cache server
CACHED_DOC_COUNT		A group that returns information about documents in the cache
	CURRENT	The total number of documents currently stored in the cache. CURRENT returns the aggregate of the owned and on-demand documents
OWNED	CURRENT	A subgroup of CACHED_DOC_COUNT The number of owned documents currently stored in the cache
DEMAND	CURRENT	A subgroup of CACHED_DOC_COUNT The number of on-demand documents currently stored in the cache

Table C–7 (Cont.) Cache Information Group

GROUP Name	ENTRY Name	Description
CACHED_DOC_SIZE		A group that returns information about the size of documents in the cache
	CURRENT	The size, in bytes, of the documents currently stored in the cache. CURRENT returns the aggregate of the owned and on-demand documents
OWNED	CURRENT	A subgroup of CACHED_DOC_SIZE The size, in bytes, of the owned documents currently stored in the cache
DEMAND	CURRENT	A subgroup of CACHED_DOC_SIZE The size, in bytes, of the on-demand documents currently stored in the cache
CACHEABILITY_ RULES		A group that returns information about the configured cacheability rules
RULE		A subgroup of CACHEABILITY_RULES
	INDEX	An index generated by Oracle9*i*AS Web Cache that represents the cacheability rule
	REGULAR_ EXPRESSION	The regular expression specified for the cacheability rule
	SITE_MASK	A string that is used to map multiple site names to one or more **origin server**s
CLUSTERS		A group that returns information about Oracle9*i*AS Web Cache clusters
	MEMBER_COUNT	The number of caches that are members of the cluster
	NAME	The cluster name
	CONFIG_CHECKSUM	A value generated by Oracle9*i*AS Web Cache that indicates the version of the configuration file

Table C–7 (Cont.) Cache Information Group

GROUP Name	ENTRY Name	Description
CACHE_INFO		A group that returns information about the size of the cache
	MAX_CACHE_SIZE	The maximum cache size as configured in the Resource Limits page
	ACTION_LIMIT_SIZE	Ninety percent of the maximum cache size (MAX_CACHE_SIZE)
	ALLOCATED_MEM_SIZE	The physical size of the cache, which is the amount of data memory allocated by Oracle9*i*AS Web Cache for cache storage and operation

Runtime Statistics Groups

Oracle9*i*AS Web Cache collects two types of runtime statistics:

- General runtime statistics
- Timed runtime statistics

Table C–8 lists the general runtime statistics groups and subgroups, the valid values that can be passed to the NAME attribute of the ENTRY element for the group, and a description of the attribute.

Table C–8 General Runtime Statistics Group

GROUP Name	ENTRY Name	Description
OPEN_CONNECTIONS		A group that returns information about the connections to the cache
	CURRENT	The number of current open connections to the cache
	MAX_SINCE_START	The maximum number of connections to the cache that have been open at the same time since the cache was last started
APP_SRVR_REQUEST_BACKLOG		A group that returns information about origin server request backlogs
	CURRENT	The current number of requests that the origin server is processing for Oracle9*i*AS Web Cache
	MAX_SINCE_START	The maximum number of requests that the origin server has processed for Oracle9*i*AS Web Cache since the cache was last started
SESSION_COUNT		A group that returns information about the connections to Oracle9*i*AS Web Cache.
	CURRENT	The number of current active connections that Oracle9*i*AS Web Cache has open
	MAX_SINCE_START	The maximum number of current connections that Oracle9*i*AS Web Cache had open at any one time since it was last started

Table C–8 (Cont.) General Runtime Statistics Group

GROUP Name	ENTRY Name	Description
CACHE_REDIRECT_DOC_ COUNT		A group that returns information about document redirection
	CURRENT	The number of documents currently redirected by Oracle9*i*AS Web Cache
	MAX_SINCE_ START	The maximum number of documents redirected by Oracle9*i*AS Web Cache since the cache was last started

Table C–9 lists the timed runtime statistics groups and subgroups and a description of each group or subgroup.

The following strings are valid values for the NAME attribute of the ENTRY element of all of the timed statistics listed in Table C–9:

- RECENT_PER_SEC: The average number for each second during the last ten seconds
- MAX_PER_SEC_SINCE_START: The maximum number for each second since the cache was last restarted
- AVG_PER_SEC_SINCE_START: The average number for each second since the cache was last restarted
- TOTAL_SINCE_START: The total number since the cache was last restarted
- TOTAL_SINCE_RESET: The total number since the statistics were reset
- AVG_PER_SEC_SINCE_RESET: The average since the statistics were reset

Table C–9 Timed Runtime Statistics Group

Group Name	Description
BYTES_SERVED	The ENTRY elements for this group return statistics about the bytes served by the cache.
HTTP_REQUESTS	The ENTRY elements for this group return statistics about the browser, peer cache, and ESI requests served by the cache.
HTTP_CLIENT_REQUESTS	The ENTRY elements for this group return statistics about the browser and peer cache requests served by the cache.

Table C–9 (Cont.) Timed Runtime Statistics Group

Group Name	Description
BYTES_SAVED_WITH_ COMPRESSION	The ENTRY elements for this group return statistics about the additional bytes sent to browsers if in-cache compression is turned off.
INVALIDATION_REQUESTS	The ENTRY elements for this group return statistics about the invalidation requests submitted to the cache.
INVALIDATED_OBJECTS	The ENTRY elements for this group return statistics about the objects invalidated.
HITS	The ENTRY elements for this group return statistics about the browser requests resolved by documents in the cache. Every ENTRY element in this group returns the aggregate value of the corresponding ENTRY elements in the subgroups.
FRESH_HITS	A subgroup of the group HITS The ENTRY elements for this group return statistics about the browser requests resolved by documents in the cache. Every ENTRY element in this group returns the aggregate value of the corresponding ENTRY elements in the subgroups.
FROM_OWNED_TO _CLIENT	A subgroup of the group FRESH_HITS The ENTRY elements for this group return statistics about browser requests resolved by owned documents in the cache.
FROM_OWNED_TO_PEER	A subgroup of the group FRESH_HITS The ENTRY elements for this group return statistics about browser requests resolved by retrieving owned documents from a peer cache.
FROM_DEMAND_TO_CLIENT	A subgroup of the group FRESH_HITS The ENTRY elements for this group return statistics about browser requests resolved by on-demand content.

Table C–9 (Cont.) Timed Runtime Statistics Group

Group Name	Description
STALE_HITS	A subgroup of the group HITS The ENTRY elements for this group return statistics about browser requests resolved by expired or invalidated content. Every STALE_HITS element in this group returns the aggregate value of the corresponding STALE_HITS elements in the subgroups.
FROM_OWNED_TO_CLIENT	A subgroup of the group STALE_HITS The ENTRY elements for this group return statistics about browser requests resolved by expired or invalidated owned content.
FROM_OWNED_TO_PEER	A subgroup of the group STALE_HITS The ENTRY elements for this group return statistics about browser requests resolved by expired or invalidated owned documents from a peer cache.
FROM_DEMAND_TO_CLIENT	A subgroup of the group STALE_HITS The ENTRY elements for this group return statistics about browser requests resolved by expired or invalidated on-demand content.
MISSES	The ENTRY elements for this group return statistics about cacheable and noncacheable misses. Misses are browser requests for documents that were not served by the cache. Every ENTRY element in this group returns the aggregate value of the corresponding ENTRY elements in the subgroups.
CACHEABLE_MISSES	A subgroup of the group MISSES The ENTRY elements for this group return statistics about cacheable misses. Every ENTRY element in this group returns the aggregate value of the corresponding ENTRY elements in the subgroups.
FROM_OWNED_TO_CLIENT	A subgroup of the group CACHEABLE_MISSES The ENTRY elements for this group return statistics about browser requests for cacheable owned documents that were not served by the cache.

Table C–9 (Cont.) Timed Runtime Statistics Group

Group Name	Description
FROM_OWNED_TO_PEER	A subgroup of the group CACHEABLE_MISSES The ENTRY elements for this group return statistics about browser requests for documents owned by a peer cache that were not served by the cache.
FROM_DEMAND_TO_CLIENT	A subgroup of the group CACHEABLE_MISSES The ENTRY elements for this group return statistics about cacheable misses for on-demand content.
NONCACHEABLE_MISSES	A subgroup of the group MISSES The ENTRY elements for this group return statistics about noncacheable misses Every ENTRY element in this group returns the aggregate value of the corresponding ENTRY elements in the subgroups.
FROM_OWNED_TO_CLIENT	A subgroup of the group NONCACHEABLE_MISSES The ENTRY elements for this group return statistics about noncacheable misses for owned documents.
FROM_OWNED_TO_PEER	A subgroup of the group NONCACHEABLE_MISSES The ENTRY elements for this group return statistics about noncacheable misses for documents owned by a peer cache.
FROM_DEMAND_TO_CLIENT	A subgroup of the group NONCACHEABLE_MISSES The ENTRY elements for this group return statistics about noncacheable misses for on-demand content.
OWNER_UNKNOWN	A subgroup of the group NONCACHEABLE_MISSES The ENTRY elements for this group return statistics about noncacheable misses for which the owner is unknown.
REFRESHES	The ENTRY elements for this group return statistics about the documents that the cache has refreshed from the application Web servers.
COMPRESSED_HITS	The ENTRY elements for this group return statistics about the total requests served from the cache in compressed form.
COMPRESSED_MISSES	The ENTRY elements for this group return statistics about the total requests retrieved from the application Web servers and compressed by the cache before serving.

Table C–9 (Cont.) Timed Runtime Statistics Group

Group Name	Description
ERRORS	The ENTRY elements for this group return statistics about the apology pages that the cache has served. Every ENTRY element in this group returns the aggregate value of the corresponding ENTRY elements in the subgroups.
NETWORK_ERRORS	A subgroup of the group ERRORS The ENTRY elements for this group return statistics about the apology pages that the cache has served to Web browsers due to a network error.
SITE_BUSY_ERRORS	A subgroup of the group ERRORS The ENTRY elements for this group return statistics about the apology pages that the cache has served to Web browsers due to a network or busy Web site error.
PARTIAL_PAGE_ERRORS	A subgroup of the group ERRORS The ENTRY elements for this group return statistics about the apology pages that the cache has served to Web browsers due to an HTML fragment retrieval problem for a page that supports partial page caching.

Site Information Group

The site information group provides multiple-site support. You can request statistics for all sites or for a specific site. Oracle9*i*AS Web Cache first checks to see if this group exists in the request message, then requests the statistics, ensuring that the correct statistics are returned to multiple-site environments.

Table C–10 lists the site information group, the valid values that can be passed to the NAME attribute of the ENTRY element for the group, and a description of the attribute.

Table C–10 Site Information Group

GROUP Name	ENTRY Name	Description
SITE		A group that returns information about a Web site
	ID	An index generated by Oracle9*i*AS Web Cache as an identifier of the site
	NAME	The name of the site

Origin Server Statistics Group

The origin server statistics group provides information and statistics about the origin server.

Origin server statistics can contain multiple SERVER subgroups. Each SERVER group contains origin server configuration information as well as origin server runtime statistics. The runtime statistics are similar in format to cache runtime statistics

Table C–11 lists the origin server statistics groups and subgroups, the valid values that can be passed to the NAME attribute of the ENTRY element for the group, and a description of the attribute.

Table C–11 Origin Server Statistics Group

GROUP Name	ENTRY Name	Description
APP_SRVR_STATS		A group that returns information about origin servers
SERVER		A subgroup of the group APP_SRVR_STATS
	HOSTNAME	The name of the host on which the origin server is running
	PORT	The port number from which the origin server is listening for Oracle9*i*AS Web Cache requests
	IS_PROXY	Whether or not the origin server is a proxy server. Valid values are "NO" and "YES"
	STATUS	The status of the origin server
	SECONDS_SINCE_STATUS_CHANGE	The number of seconds since the status of the origin server changed
REQUESTS		A subgroup of the group SERVER
	RECENT_PER_SECOND	The average number of requests served for each second during the last ten seconds
	MAX_PER_SEC_SINCE_START	The maximum number of requests served for each second since the origin server was last restarted
	AVG_PER_SEC_SINCE_START	The average number of requests served for each second since the origin server was last restarted
	TOTAL_SINCE_START	The total number of requests served since the origin server was last restarted.
	TOTAL_SINCE_RESET	The total number of requests served since the statistics were reset
	AVG_PER_SEC_SINCE_RESET	The average number of requests served for each second since the statistics were reset

Table C–11 (Cont.) Origin Server Statistics Group

GROUP Name	ENTRY Name	Description
LATENCY		A subgroup of the group SERVER
	RECENT_PER_SECOND	The average number of seconds, in the last 10-second interval, used to process requests for Oracle9*i*AS Web Cache
	MAX_PER_SEC_SINCE_START	The maximum number of seconds used to process requests for Oracle9*i*AS Web Cache since the origin server started
	AVG_PER_SEC_SINCE_START	The average number of seconds used to process requests for Oracle9*i*AS Web Cache since the origin server started
	TOTAL_SINCE_START	The total number of seconds used to process requests for Oracle9*i*AS Web Cache since the origin server started
	TOTAL_SINCE_RESET	The total number of seconds used to process requests for Oracle9*i*AS Web Cache since the statistics were reset
	AVG_PER_SEC_SINCE_RESET	The average number of seconds used to process requests for Oracle9*i*AS Web Cache since the statistics were reset
ACTIVE_SESSIONS		A subgroup of the group SERVER
	CURRENT	The number of current active connections from Oracle9*i*AS Web Cache that the origin server has open
	MAX_SINCE_START	The maximum number of active connections from Oracle9*i*AS Web Cache that the origin server has had open at any one time
OPEN_CONNECTIONS		A subgroup of the group SERVER
	CURRENT	The number of current connections from Oracle9*i*AS Web Cache that the origin server has open

Table C–11 (Cont.) Origin Server Statistics Group

GROUP Name	ENTRY Name	Description
	MAX_SINCE_START	The maximum number of connections from Oracle9*i*AS Web Cache that the origin server has had open at any one time

URL Statistics Group

The URL statistics group returns the URLs of the most popular documents. You specify the number of URLs to be returned. This is the information returned as the Most Popular Documents in the Cache Contents page (**Administration** > **Monitoring** > **Cache Contents**) of Oracle9*i*AS Web Cache Manager.

The URL statistics group can contain multiple URL subgroups.

Table C–12 lists the URL statistics groups and subgroups, the valid values that can be passed to the NAME attribute of the ENTRY element for the group, and a description of the attribute.

Table C–12 URL Statistics Group

GROUP Name	PARAM or ENTRY Name	Description
URL_STATS		A group that returns information about the most popular documents in the cache.
	OBJECT_COUNT (PARAM)	The number of the most popular URLs to be returned. You must supply a number to the VALUE attribute.
URL		A subgroup of the group URL_STATS
	URLNAME (ENTRY)	The URL of the request.
	SCORE (ENTRY)	An internally generated, relative value that indicates the popularity of a URL
	CACHABILITYRULE (ENTRY)	The index to the cacheability rule that triggered the document to be cached
	SIZE (ENTRY)	The size of the document represented by the URL

Query Methods

To retrieve Oracle9*i*AS Web Cache statistics, you send a POST message to the Oracle9*i*AS Web Cache statistics port. By default, the statistics port is 4002.

Each request must include the authentication header as part of the message. The following example shows the authentication header:

```
POST / HTTP/1.0
Authorization: BASIC < base64 encoding of administrator:administrator_password>
content-length: #bytes
```

In the example, #bytes refers to the size, in bytes, of the body of the statistics request.

The body of a statistics request must begin with the following:

```
<?xml version="1.0"?>
<!DOCTYPE WCSTATS SYSTEM "internal:///wcstats.dtd">

<WCSTATS DTD_VERSION="1.0">
```

If the request XML message contains the ENTRY element, the response will return the value of that ENTRY. If the message contains the GROUP element without any children, it will return all ENTRY values for the group as well as all ENTRY values for the subgroup.

Examples

The following examples illustrate XML request and response messages.

Figure C–6 shows the request and response messages that retrieve the URLs of 50 of the most popular objects in the cache. It uses the `GROUP ID, 124`, rather than the `GROUP NAME, URL_STATS`.

Figure C–6 Obtaining the URLs of the Most Popular Documents

The following code shows the request:

```
<?xml version="1.0"?>
<!DOCTYPE WCSTATS SYSTEM "internal:///wcstats.dtd">

<WCSTATS DTD_VERSION="1.0">
<GROUP NAME="124"/>
   <!-- NAME="URL_STATS" -->
   <PARAM NAME="OBJECT_COUNT" VALUE="50"/>
</WCSTATS>
```

The following code shows the response:

```
<?xml version="1.0"?>
<!DOCTYPE WCSTATS SYSTEM "internal:///wcstats.dtd"/>

<WCSTATS  DTD_VERSION="1.0">
<GROUP NAME="124">
 <PARAM NAME="OBJECT_COUNT" VALUE="50"/>
 <GROUP NAME="URL"/>
   <ENTRY NAME="NAME" VALUE="/sitename:port/admin/images/headers/maglass.gif/"/>

   <ENTRY NAME="SCORE" VALUE="99"/>
   <ENTRY NAME="CACHABILITYRULE" VALUE="1"/>
   <ENTRY NAME="SIZE" VALUE="1037"/>
 </GROUP>
.
.
.
</GROUP>
</WCSTATS>
```

Figure C–7 shows the request and response messages that retrieve the number of objects recently invalidated for each second. The example uses the `GROUP ID`, `107`, rather than the `GROUP NAME`, `INVALIDATED_OBJECTS`.

Figure C–7 Obtaining the Number of Invalidated Objects

The following code shows the request:

```
<?xml version="1.0"?>
<!DOCTYPE WCSTATS SYSTEM "internal:///wcstats.dtd">

<WCSTATS  DTD_VERSION="1.0">
<GROUP NAME="107"> <!-- NAME="INVALIDATED_OBJECTS" -->
  <ENTRY NAME="RECENT_PER_SEC"/>
</GROUP>
</WCSTATS>
```

The following code shows the response:

```
<?xml version="1.0"?>
<!DOCTYPE WCSTATS SYSTEM "internal:///wcstats.dtd">

<WCSTATS  DTD_VERSION="1.0">
<GROUP NAME="107" NAME="INVALIDATED_OBJECTS">
 <ENTRY NAME="RECENT_PER_SEC" VALUE="50"/>
</GROUP>
</WCSTATS>
```

Figure C–8 shows the request and response messages that retrieve all statistics for the group INVALIDATED_OBJECTS. The example uses the GROUP ID, 107, rather than the GROUP NAME, INVALIDATED_OBJECTS. Because the request contains only the GROUP element, Oracle9*i*AS Web Cache returns all statistics in the group.

Figure C–8 Obtaining All Statistics for Invalidated Objects

The following code shows the request:

```
<?xml version="1.0"?>
<!DOCTYPE WCSTATS SYSTEM "internal:///wcstats.dtd">

<WCSTATS DTD_VERSION="1.0">
<GROUP NAME="107"/>
</WCSTATS>
```

The following code shows the response:

```
<?xml version="1.0"?>
<!DOCTYPE WCSTATS SYSTEM "internal:///wcstats.dtd">

<WCSTATS DTD_VERSION="1.0">
<GROUP NAME="107"> <!-- INVALIDATED_OBJECTS -->
 <ENTRY NAME="RECENT_PER_SEC" VALUE="50"/>
 <ENTRY NAME="MAX_SINCE_START" VALUE="100"/>
 <ENTRY NAME="AVG_SINCE_START" VALUE="36"/>
 <ENTRY NAME="TOTAL_SINCE_START" VALUE="1000"/>
 <ENTRY NAME="TOTAL_SINCE_RESET" VALUE="500"/>
 <ENTRY NAME="AVG_SINCE_RESET" VALUE="30"/>
</GROUP>
</WCSTATS>
```

Complete Statistics Template

Figure C–9 lists a complete template for all elements in the statistics DTD.

Figure C–9 Complete Statistics Template

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

<WCSTATS>
<!DOCTYPE WCSTATS SYSTEM "internal:///wcstats.dtd">
<WCSTATS DTD_VERSION="1.0">

<!-- Cache_Information -->
<GROUP NAME="TIME">
  <ENTRY NAME="CACHE_START_TIME" VALUE=""/>
  <ENTRY NAME="STATS_RESET_TIME" VALUE=""/>
  <ENTRY NAME="LAST_MODIFIED_TIME" VALUE=""/>
</GROUP>

<GROUP NAME="PID">
  <ENTRY NAME="CACHE_PROCESS" VALUE=""/>
</GROUP>

<GROUP NAME="CACHED_DOC_COUNT">
  <ENTRY NAME="CURRENT" VALUE=""/>
  <GROUP NAME="OWNED">
    <ENTRY NAME="CURRENT" VALUE=""/>
  </GROUP>
  <GROUP NAME="DEMAND">
    <ENTRY NAME="CURRENT" VALUE=""/>
  </GROUP>
</GROUP>
<GROUP NAME="CACHED_DOC_SIZE">
  <ENTRY NAME="CURRENT" VALUE=""/>
  <GROUP NAME="OWNED">
    <ENTRY NAME="CURRENT" VALUE=""/>
  </GROUP>
  <GROUP NAME="DEMAND">
    <ENTRY NAME="CURRENT" VALUE=""/>
  </GROUP>
</GROUP>
```

```
<GROUP NAME="CACHEABILITY_RULES">
  <GROUP NAME="RULE">
    <ENTRY NAME="INDEX" VALUE=""/>
    <ENTRY NAME="REGULAR_EXPRESSION" VALUE=""/>
    <ENTRY NAME="SITE_MASK" VALUE=""/>
  </GROUP>
    .
    .
    .
</GROUP>
<GROUP NAME="CLUSTERS">
  <ENTRY NAME="MEMBER_COUNT" VALUE=""/>
  <ENTRY NAME="NAME" VALUE=""/>
  <ENTRY NAME="CONFIG_CHECKSUM" VALUE=""/>
</GROUP>

<GROUP NAME="CACHE_INFO">
  <ENTRY NAME="MAX_CACHE_SIZE" VALUE=""/>
  <ENTRY NAME="ACTION_LIMIT_SIZE" VALUE=""/>
  <ENTRY NAME="ALLOCATED_MEM_SIZE" VALUE=""/>
</GROUP>

<!-- General Runtime Statistics -->
<GROUP NAME="OPEN_CONNECTIONS">
  <ENTRY NAME="CURRENT" VALUE=""/>
  <ENTRY NAME="MAX_SINCE_START" VALUE=""/>
</GROUP>

<GROUP NAME="APP_SRVR_REQUEST_BACKLOG">
  <ENTRY NAME="CURRENT" VALUE=""/>
  <ENTRY NAME="MAX_SINCE_START" VALUE=""/>
</GROUP>
<GROUP NAME="SESSION_COUNT">
  <ENTRY NAME="CURRENT" VALUE=""/>
  <ENTRY NAME="MAX_SINCE_START" VALUE=""/>
</GROUP>
<GROUP NAME="CACHE_REDIRECT_DOC_COUNT">
  <ENTRY NAME="CURRENT" VALUE=""/>
  <ENTRY NAME="MAX_SINCE_START" VALUE=""/>
</GROUP>
```

```
<!-- Timed Runtime Statistics -->
<GROUP NAME="BYTES_SERVED">
  <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
  <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
</GROUP>

<GROUP NAME="HTTP_REQUESTS">
  <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
  <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
</GROUP>
<GROUP NAME="HTTP_CLIENT_REQUESTS">
  <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
  <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
</GROUP>

<GROUP NAME="BYTES_SAVED_WITH_COMPRESSION">
  <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
  <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
</GROUP>

<GROUP NAME="INVALIDATION_REQUESTS">
  <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
  <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
</GROUP>
```

```
<GROUP NAME="INVALIDATED_OBJECTS">
  <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
  <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
</GROUP>
<GROUP NAME="HITS">
<ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
<ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
<ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
<ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
<ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
<ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>

<GROUP NAME="FRESH_HITS">
  <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
  <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
  <GROUP NAME="FROM_OWNED_TO_CLIENT">
    <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
    <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
  </GROUP>
  <GROUP NAME="FROM_OWNED_TO_PEER">
    <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
    <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
  </GROUP>
  <GROUP NAME="FROM_DEMAND_TO_CLIENT">
    <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
    <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
  </GROUP>
</GROUP>
```

```
<GROUP NAME="STALE_HITS">
  <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
  <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
  <GROUP NAME="FROM_OWNED_TO_CLIENT">
    <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
    <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
  </GROUP>
  <GROUP NAME="FROM_OWNED_TO_PEER ">
    <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
    <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
  </GROUP>
  <GROUP NAME="FROM_DEMAND_TO_CLIENT">
    <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
    <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
  </GROUP>
</GROUP>
</GROUP>
<GROUP NAME="MISSES">
<ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
<ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
<ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
<ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
<ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
<ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
```

```
<GROUP NAME="CACHEABLE_MISSES">
  <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
  <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
  <GROUP NAME="FROM_OWNED_TO_CLIENT">
    <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
    <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
  </GROUP>
  <GROUP NAME="FROM_OWNED_TO_PEER">
    <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
    <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
  </GROUP>
  <GROUP NAME="FROM_DEMAND_TO_CLIENT">
    <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
    <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
  </GROUP>
</GROUP>
<GROUP NAME="NONCACHEABLE_MISSES">
  <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
  <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
  <GROUP NAME="FROM_OWNED_TO_PEER">
    <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
    <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
  </GROUP>
```

```
    <GROUP NAME="FROM_OWNED_TO_CLIENT">
      <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
      <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
      <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
      <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
      <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
      <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
    </GROUP>
    <GROUP NAME="FROM_DEMAND_TO_CLIENT">
      <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
      <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
      <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
      <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
      <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
      <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
    </GROUP>
    <GROUP NAME="OWNER_UNKNOWN">
      <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
      <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
      <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
      <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
      <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
      <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
    </GROUP>
  </GROUP>
  </GROUP>
  <GROUP NAME="REFRESHES">
    <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
    <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
  </GROUP>
  <GROUP NAME="COMPRESSED_HITS">
    <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
    <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
    <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
    <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
  </GROUP>
```

```
<GROUP NAME="COMPRESSED_MISSES">
  <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
  <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
</GROUP>
<GROUP NAME="ERRORS">
<ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
<ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
<ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
<ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
<ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
<ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
<GROUP NAME="NETWORK_ERRORS">
  <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
  <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
</GROUP>
<GROUP NAME="SITE_BUSY_ERRORS">
  <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
  <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
</GROUP>
<GROUP NAME="PARTIAL_PAGE_ERRORS">
  <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
  <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
  <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
  <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
</GROUP>
</GROUP>
```

```
<!-- Multisite Statistics -->
<GROUP NAME="SITE">
  <ENTRY NAME="ID" VALUE=""/>
  <ENTRY NAME="NAME" VALUE=""/>
</GROUP>

<!-- Origin Server Statistics-->
<GROUP NAME="APP_SRVR_STATS">
  <GROUP NAME="SERVER">
    <ENTRY NAME="HOSTNAME" VALUE="">
    <ENTRY NAME="PORT" VALUE="">
    <ENTRY NAME="IS_PROXY" VALUE="">
    <ENTRY NAME="STATUS" VALUE="">
    <ENTRY NAME="SECONDS_SINCE_STATUS_CHANGE" VALUE=""/>
    <GROUP NAME="REQUESTS">
      <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
      <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
      <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
      <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
      <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
      <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
    </GROUP>
    <GROUP NAME="LATENCY">
      <ENTRY NAME="RECENT_PER_SEC" VALUE=""/>
      <ENTRY NAME="MAX_PER_SEC_SINCE_START" VALUE=""/>
      <ENTRY NAME="AVG_PER_SEC_SINCE_START" VALUE=""/>
      <ENTRY NAME="TOTAL_SINCE_START" VALUE=""/>
      <ENTRY NAME="TOTAL_SINCE_RESET" VALUE=""/>
      <ENTRY NAME="AVG_PER_SEC_SINCE_RESET" VALUE=""/>
    </GROUP>
    <GROUP NAME="ACTIVE_SESSIONS">
      <ENTRY NAME="CURRENT" VALUE=""/>
      <ENTRY NAME="MAX_SINCE_START" VALUE=""/>
    </GROUP>
    <GROUP NAME="OPEN_CONNECTIONS">
      <ENTRY NAME="CURRENT" VALUE=""/>
      <ENTRY NAME="MAX_SINCE_START" VALUE=""/>
    </GROUP>
  </GROUP>
  ...
</GROUP>
```

```
<!-- URL Statistics -->
<GROUP NAME="URL_STATS">
  <PARAM NAME="OBJECT_COUNT" VALUE="50"/>
  <GROUP NAME="URL">
    <ENTRY NAME="URLNAME" VALUE=""/>
    <ENTRY NAME="SCORE" VALUE=""/>
    <ENTRY NAME="CACHABILITYRULE" VALUE=""/>
    <ENTRY NAME="SIZE" VALUE=""/>
  </GROUP>
  .
  .
  .
</GROUP>

</WCSTATS>
```

D

Edge Side Includes Language

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

This appendix contains these topics:

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 Oracle9*i*AS 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 Oracle9*i*AS 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 7-43
- "Configuring Caching Attributes in Response Headers" on page 7-66
- `http://www.edge-delivery.org` for the ESI language release 1.0 specification

The following topics provide an overview of ESI usage:

- ESI Language Elements Supported
- Syntax Rules
- Nesting Elements
- Variable Expressions
- Exceptions and Errors
- Enabling ESI

ESI Language Elements Supported

ESI supports the language elements listed in Table D–1.

Table D–1 Language Elements Supported in ESI Release

ESI Language Element	See Also
`<esi:include>` tag	"ESI include Tag" on page D-14
`src` attribute	"ESI include Tag" on page D-14
`onerror` attribute	"ESI include Tag" on page D-14
`<esi:inline>` tag	"ESI inline Tag" on page D-19
`<esi:choose>` | `<esi:when>` | `<esi:otherwise>` tags	"ESI choose | when | otherwise Tags" on page D-23
`<esi:try>` | `<esi:attempt>` | `<esi:accept>` tags	"ESI try | attempt | except Tags" on page D-27
`<esi:comment>` tag	"ESI comment Tag" on page D-28
`<esi:remove>` tag	"ESI remove Tag" on page D-29
`<esi:vars>` tag	"ESI vars Tag" on page D-31
`<!--esi...--->` tag	"ESI <!--esi--->Tag" on page D-30

In addition to the ESI elements, Oracle supplies the proprietary language elements listed in Table D–2.

Table D–2 Oracle Language Elements

Oracle Language Elements	See Also
`<esi:environment>` tag	"ESI environment Tag" on page D-14
`<esi:include>` tag	"ESI include Tag" on page D-14
`name` attribute	"ESI include Tag" on page D-14
`max-age` attribute	"ESI include Tag" on page D-14
`timeout` attribute	"ESI include Tag" on page D-14
`<esi:request_header>` element	"ESI include Tag" on page D-14
`<esi:request_body>` element	"ESI include Tag" on page D-14

Syntax Rules

ESI elements and attributes adhere to XML syntax but can be embedded in other documents such as HTML or XML documents. When Oracle9*i*AS 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 are case sensitive.

  They are generally lowercase.

- Supported CGI environment variables are case sensitive.

  They are generally 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'" >
            <img src="/img/TheImage.gif">
          </esi:when>
          <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–3 on page D-6 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 request body or the URL. Variables are only interpreted when enclosed within ESI tags.

Table D–3 ESI-Supported Variables

Variable Name	HTTP Header Field	Substructure Type/Variable Type	Description	Example

`$(HTTP_ACCEPT_LANGUAGE{`*`language`*`})`	`Accept-Language` request-header field
	Specifies the set of languages that are preferred as a response. The language is used as the key.
		List / Boolean	Specifies the language to use as the key and evaluates to to the language specified in the HTTP request header	Variable Setting:
				`$(HTTP_LANGUAGE{en-gb})`
				HTTP Request Header Contains:
				`Accept_Language:en-gb`
				Result:
				Evaluates to `en-gb`.

`$(HTTP_COOKIE{`*`cookie`*`})`	`Set-Cookie` response-header field or `Cookie` request-header field
	Specifies cookie name and value pairs.A cookie name is used as the key.
	If the `Cookie` request-header and `Set-Cookie` response-header have different values for the same cookie name, then the name value pair from the `Set-Cookie` response header is used.
		Dictionary / String	Specifies the cookie name to use as the key and returns that cookie's value	Variable Setting:
				`$(HTTP_COOKIE{visits})`
				HTTP Request Header Contains:
				`Cookie:visits=42`
				Result:
				Returns a value of 42.

`$HTTP_HEADER{`*`header`*`})`	Any HTTP request header	Dictionary / String	Specifies an HTTP request header name to use as the key and returns that header's value	Variable Setting:
				`$(HTTP_HEADER{Referer})`
				HTTP Request Header Contains:
				`Referer: http://www.company.com:80`
				Result:
				Returns a value of `http://www.company.com:80`

Table D–3 (Cont.) ESI-Supported Variables

Variable Name	HTTP Header Field	Substructure Type/Variable Type	Description	Example

`$HTTP_HOST`	`Host` request-header field
	Specifies the host name and port number of the resource. Port 80 is the default port number.
		Not Applicable / String	Returns the value of the `HOST` header	Variable Setting:
				`$(HTTP_HOST)`
				HTTP Request Header Contains:
				`Host:http://www.company.com:80`
				Result:
				Returns a value of `http://www.company.com:80`

`$HTTP_REFERER`	`Referer` request-header field
	Specifies the URL of the reference resource
		Not Applicable / String	Returns the value of the `REFERER` header	Variable Setting:
				`$(HTTP_REFERER)`
				HTTP Request Header Contains:
				`Referer:http://www.company.com:80`
				Result:
				Returns `http://www.company.com:80`

`$(HTTP_USER_AGENT{browser})`
`$HTTP_USER_AGENT{version})`
`$HTTP_USER_AGENT{os})`
	`User-Agent` request-header field
	Specifies the Web browser type, browser version, or operating system that initiated the request.
		Dictionary / String	Specifies one of three keys: `browser` for browser type, `version` for browser version, and `os` for operating system	Variable Setting:
				`$(HTTP_USER_AGENT{browser})`
				HTTP Request Header Contains:
				`User-Agent:Mozilla/4.0 (compatible, MSIE 5.5, Windows NT 4.0)`
				Result:
				Returns `Mozilla`
				`$(HTTP_USER_AGENT{version})`
				Returns `4.0`.
				`$(HTTP_USER_AGENT{os})`
				Returns `Windows NT 4.0`.

Table D–3 (Cont.) ESI-Supported Variables

Variable Name	HTTP Header Field	Substructure Type/Variable Type	Description	Example
`$(QUERY_STRING{`*`parameter`*`})`	Not Applicable	Dictionary / String	Given a parameter name in a query string, returns the value of the parameter without URL encoding. The query string can be in an URL or a request body. **See Also:** `http://rfc.net/rfc1738.html` for further information about URL encoding.	Variable Setting: `$(QUERY_STRING{CEO})` Result: Returns the value of `fullname` decoded. In this example, `CEO` returns a value of `Jane Doe.`

`$(QUERY_STRING)`	Not Applicable	Not Applicable / String	Specifies to return the entire query string encoded	Variable Setting:
				`$(QUERY_STRING)`
				Result:
				Returns the entire query string encoded:
				`CEO=Jane%20Doe&CFO=John%20Doe`

Table D–3 (Cont.) ESI-Supported Variables

Variable Name	HTTP Header Field	Substructure Type/Variable Type	Description	Example

`$(QUERY_STRING_ENCODED{`*`parameter`*`})`	Not Applicable	Dictionary / String	Given a parameter name in a query string, returns the value of the parameter with URL encoding. The query string can be in an URL or a request body.	Variable Setting:
				`$(QUERY_STRING{CEO})`
				Result:
				Returns the value of `fullname` encoded:
				`Jane%20Doe`

`$(QUERY_STRING_ENCODED)`	Not Applicable	Not Applicable / String	The same as `$(QUERY_STRING)`	Variable Setting:
				`$(QUERY_STRING_ENCODED)`
				Result:
				Returns the entire query string encoded:
				`CEO=Jane%20Doe&CFO=John%20Doe`

`$(QUERY_STRING_DECODED{`*`parameter`*`})`	Not Applicable	Dictionary / String	The same as `$(QUERY_STRING{`*`parameter`*`})`	Variable Setting: `$(QUERY_STRING_DECODED{CEO})` Result: Returns the value of `fullname` decoded. In this example, `CEO` returns a value of `Jane Doe`.

See Also: "ESI environment Tag" on page D-21 for instructions on including custom variables

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–3 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 and optional. If a key is not specified, then the environment variable returns the whole content of the environment fragment. Oracle Corporation advises specifying an environment variable without a key only for testing whether the environment is empty. In the following ESI markup, `$(logindata)` is a variable that is evaluated against a `null` value.

```
<esi:choose>
  <esi:when test="$(logindata) != null">
    <esi:include src=/login/$(logindata)"/>
 </esi:when>
 <esi:otherwise>
   <esi:include src=/login/guest.html"/>
 <esi:otherwise>
</esi:choose>
```

Variables identified with a substructure type of Dictionary in Table D–3 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–3 return a boolean value depending on whether the requested value is present.

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 Oracle9*i*AS 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:

- Apology Page
- ESI Language Control

Apology Page

The first mechanism is to use an apology page in place of an included fragment.

To configure an apology page:

1. Create an apology page in `$ORACLE_HOME/webcache/docs` on UNIX and *`ORACLE_HOME`*`\webcache\docs` on Windows.
2. Specify the file name of the apology page in the Apology Page (**General Configuration** > **Apology Pages**) of Oracle9*i*AS Web Cache Manager.

> **See Also:** Step 4 in"Task 10: Configure Web Site Settings" on page 6-26

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

The `onerror` attribute is used before the `try` | `attempt` | `except` block. If the `try` | `attempt` | `except` block does not exist, then the exception handling is propagated to the parent or template page. The parent or template page will use the apology page, `onerror` attribute, or `try` | `attempt` | `except` block to handle the error.

> **See Also:**
>
> - "ESI include Tag" on page D-14
> - "ESI try | attempt | except Tags" on page D-27

Enabling ESI

To enable Oracle9*i*AS 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 inline Tag
- ESI environment 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.

> **See Also:** "Fragmentation with the Inline and Include Tags" on page 2-32 for a comparison of <esi:inline> and <esi:include> usage

Syntax

In this form, <esi:include> does not have a closing </esi:include>:

```
<esi:include src="URL_fragment"
[max-age="expiration_time [+ removal_time]]" [method="GET|POST"]
[onerror="continue"] [timeout="fetch_time"]/>
```

In this form with elements, <esi:include> has a closing </esi:include>:

```
<esi:include src="URL_fragment"
[max-age="expiration_time [+ removal_time]"] [method="GET|POST"]
[onerror="continue"] [timeout="fetch_time"]>
  [<esi:request_header name="request_header" name="request_header"
value="value"/>]
  [<esi:request_body value="value"/>]
</esi:include>
```

Attributes

- src—Specifies the fragment to fetch. The fragment can be a file referenced by a URL or it can include variables. You can specify an XML fragment as long as the XML file fragment includes the following code at the beginning of the file:

  ```
  <?xml version="1.0"?>
  <?xml-stylesheet type="text/xml" href="stylesheet.css"?>
  ```

  Oracle9*i*AS Web Cache uses XSL Transformations (XSLT) to transform the XML into HTML using a stylesheet. The stylesheet maps XML formats to HTML formats.

  > **See Also:** http://www.xslt.com/ for complete information about XSLT

- `max-age`—Specifies the time, in seconds, to expire the fragment, and optionally, specifies the time, in seconds, to remove the fragment after the expiration time. Use this attribute if the template page has a higher tolerance for stale fragments than specified by the time-to-live parameters in fragment responses.
- `method`—Specifies the **HTTP request method** of the document(s). Valid values are `GET` or `POST`.
- `onerror`—Specifies that if the fetch failed on the `src` object to ignore the ESI tag and serve the page
- `timeout`—Specifies the time, in seconds, for the fragment to be fetched. If the fragment has not been fetched within the time interval, then the fetch is aborted.

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 Oracle9*i*AS Web Cache is near the application Web server, the `alt` tag cannot be processed in a useful manner. Therefore, Oracle9*i*AS Web Cache ignores the `alt` attribute.

Elements

- `request_body`—Specifies the HTTP request body of the fragment
- `request_header`—Specifies an HTTP request header field and value for Oracle9*i*AS Web Cache to use. You can specify multiple HTTP request headers. When this attribute is specified, all request headers from the parent fragment or template page are ignored.

Syntax Usage

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

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

- The attributes do not need to be in a particular order

- The `src` support both HTTP and HTTPS. Oracle9*i*AS Web Cache permits the template and fragments to use different protocols. Take note of the following:
  - If the `src` attribute specifies a fragment's relative path, such as `src="/PersonalizedGreeting"`, then the template's protocol is used.
  - If the protocol used in the `src` attribute does not match the protocol specified in the Site to Server Mapping page (**General Configuration** > **Site to Server Mapping**) of Oracle9*i*AS Web Cache Manager, then Oracle9*i*AS Web Cache uses the protocol configured for the origin server in the Site to Server Mapping page. Oracle9*i*AS Web Cache also reports the following warning message to the event log:

    Date `Warning: ESI Include protocol does not match Origin Server protocol: Origin Server Protocol=`*protocol* `URL=`*URL*

    For example, if the template page is configured with `<esi:include src="https://www.company.com/gifs/frag1.gif"/>` and the Site to Server mapping specifies HTTP for the origin server, then `http://www.company.com/gifs/frag1.gif` is used and the following message appears in the event log:

    ```
    11/Jan/2002:19:25:59 +0000 Warning: ESI Include protocol does not match
    Origin Server protocol: Origin Server Protocol=http
    URL=https://www.company.com/gifs/frag1.gif
    ```

- Do not specify more than one `request_body` element
- You can have zero or more `request_body` elements

  Use multiple `request_header` elements to specify multiple HTTP request header fields:

  ```
  <esi:include src="URL_fragment"
  max-age="expiration_time [+ removal_time]"[method="GET|POST"]
  onerror="continue" timeout="fetch_time">
    <esi:request_header name="request_header" value="value"/>
    <esi:request_header name="request_header" value="value"/>
  </esi:include>
  ```

Usage

The `<esi:include>` tag instructs Oracle9*i*AS 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, then 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 Oracle9*i*AS Web Cache cannot fetch the `src`, then Oracle9*i*AS Web Cache tries to return an apology page error. If Oracle9*i*AS Web Cache cannot use the apology page and `onerror="continue"` is specified, then Oracle9*i*AS Web Cache ignores the `<esi:include>` tag. If `onerror="continue"` is not specified, then Oracle9*i*AS Web Cache looks for the `try` | `attempt` | `except` block. If the `try` | `attempt` | `except` block does not exist, then the exception handling is propagated to the parent or template page. The parent page will use the apology page, `onerror` attribute, or `try` | `attempt` | `except` block to handle the error. The template page will return an internal error.

If both the `max-age` control directive is set in the `Surrogate-Control` response-header field and the `max-age` attribute are set, then Oracle9*i*AS Web Cache uses the longest maximum age of the two. Oracle Corporation recommends setting the `max-age` attribute to a longer time than the `max-age` control directive. Use the `max-age` attribute to increase cache hits by serving fragments stale until the removal time. `max-age=infinity` specifies that the document never expires.

If `method` is not set, then `GET` is assumed. However, if the `request_body` element is set, then `POST` is assumed.

Oracle9*i*AS Web Cache generates the following HTTP request headers for all fragment requests:

- `Host:`*`host`*`:`*`port`*
- `Content-Length:`*`size`* (for HTTP `POST` body documents)
- `Surrogate-Capability:orcl="ORAESI/9.0.2 ESI/1.0"`
- `Connection:Keep-Alive|Close`

The `request_header` element enables you to control HTTP header other than these. Do not specify these HTTP request headers as `request_header` attributes, as a conflict can affect the operation of Oracle9*i*AS Web Cache.

If no `request_header` elements are specified, then Oracle9*i*AS Web Cache uses other request headers from the parent fragment or template page.

> **See Also:** "Fragmentation with the Inline and Include Tags" on page 2-32 for a comparison of `<esi:inline>` and `<esi:include>` usage

Examples

The following ESI markup includes a file named `frag1.htm`. The fragment must be fetched within 60 seconds. If the fetch fails, then Oracle9*i*AS Web Cache ignores the includes and serves the page. If the fetch succeeds, then Oracle9*i*AS Web Cache includes the fragment. Oracle9*i*AS Web Cache expires the fragment after five minutes, and removes it after another eight minutes.

```
<esi:include src="/frag1.htm" timeout="60" maxage="300+480" onerror="continue"/>
```

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

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

The following ESI output includes a personalized greeting, a `Cookie` HTTP request header, and a HTTP request body that includes the date:

```
<esi:include src="/PersonalGreeting"
  <esi:request_header name="Cookie" value="pname=Scott Tiger"/>
  <esi:request_body value="day=05, month=10, year=2001"/>
</esi:include>
```

> **See Also:** "Example Portal Site Implementation" on page 7-46 for an extended example of `<esi:include>` usage

ESI inline Tag

The <esi:inline> tag marks a fragment as a separately cacheable fragment, embedded in the HTTP response of another object. Oracle9*i*AS Web Cache stores and assembles these fragments independently as <esi:include> fragments.

> **See Also:** "Fragmentation with the Inline and Include Tags" on page 2-32 for a comparison of <esi:inline> and <esi:include> usage

Syntax

```
<esi:inline name="URL" fetchable="yes|no"
[max-age="expiration_time [+ removal_time]"] [timeout="fetch_time"]
Embedded HTML code
</esi:inline>
```

Attributes

- name—Specifies a unique name for the fragment in URL format
- fetchable—yes specifies that the URL can be used to independently fetch the fragment. no specifies that the fragment can only be generated as a response for another object.
- max-age—Specifies the time, in seconds, to expire the fragment, and optionally, specifies the time, in seconds, to remove the fragment after the expiration time. Use this attribute if the template page has a higher tolerance for stale fragments than specified by the time-to-live parameters in fragment responses.
- timeout—Specifies the time, in seconds, for the fragment to be fetched. If the fragment has not been fetched within the time interval, the fetch is aborted.

Usage

Some inline fragments are only delivered as part of an HTTP response for another object. These are not independently fetchable by Oracle9*i*AS Web Cache the way `<esi:include>` fragments are. When a non-fetchable fragment is needed by Oracle9*i*AS Web Cache, Oracle9*i*AS Web Cache must request the object from which the inline fragment was extracted.

When a non-fetchable `<esi:inline>` fragment is not found in the cache, Oracle9*i*AS Web Cache re-fetches the fragment's parent template. This behavior implies that the parent cannot be another non-fetchable `<esi:inline>` fragment. If the parent is an `<esi:inline>` non-fetchable fragment, then the response is returned to the browser is undefined.

See Also:

- "Using Inline for Non-Fetchable Fragmentation" on page 2-32
- "Using Inline for Fetchable Fragmentation" on page 2-34
- "ESI include Tag" on page D-14 for usage notes on the `maxage` attribute

Example

The following ESI output embeds finance headlines:

```
<esi:inline name="/Top_News_Finance">
Latest News for finance
<TABLE>
  <TR>
   Blue-Chip Stocks Cut Losses; Nasdaq Up MO
    Stocks Fall at Opening After Plane Crash New York Times
    French rig factory with explosives New York Times
    Volkswagen faces Brazil strike CNN Europe
    Airbuss reliability record BBC
  </TR>
</TABLE>
</esi:inline>
```

See Also: "Example Portal Site Implementation" on page 7-46 for an extended example of `<esi:inline>` usage

ESI environment Tag

The <esi:environment> tag enables you to include custom environment variables from included fragments. Once included, these variables can then be used with the other ESI tags.

Syntax

```
<esi:environment src="environment_URL" name="environment_name"
[max-age="expiration_time [+ removal_time]]" [method="GET|POST"]
[onerror="continue"] [timeout="fetch_time"]/>
```

In this form, <esi:environment> does not have a closing </esi:environment>.

```
<esi:environment src="environment_URL" name="environment_name"
[max-age="expiration_time [+ removal_time]"] [method="GET|POST"]
[onerror="continue"] [timeout="fetch_time"]>
  [<esi:request_header name="request_header" name="request_header"
value="value"/>]
  [<esi:request_body value="value"/>]
</esi:environment>
```

In this form with elements, <esi:environment> has a closing </esi:environment>.

Attributes

- src—Specifies the URL from which to obtain environment variables and their values. The URL requires the following XML format:

  ```
  <?xml version="1.0"?>
  <esi-environment esiversion="ORAESI/9.0.2">
    <variable_name>variable_value</variable_name>
    <variable_name>variable_value</variable_name>
  </esi-environment>
  ```

- name—Specifies the name to use to refer to the environment variable
- method—Specifies the HTTP request method of the documents to be invalidated. Valid values are GET or POST.
- max-age—Specifies the time, in seconds, to expire the XML file, and optionally, specifies the time, in seconds, to remove the XML file after the expiration time
- timeout—Specifies the time, in seconds, for the fragment to be fetched. If the fragment has not been fetched within the time interval, the fetch is aborted.

- `onerror`—Specifies that if the fetch failed on the `src` object, to ignore the ESI tag and serve the page

Elements

- `request_body`—Specifies the HTTP request body of the fragment
- `request_header`—Specifies an HTTP request header field and value for Oracle9*i*AS Web Cache to use

Usage

To use the included custom variables with other ESI tags, specify the ESI environment tag before other tags. The usage of custom variables is the same as the ESI variables.

See Also:

- "Variable Expressions" on page D-5 for usage instructions
- "ESI include Tag" on page D-14 for a usage notes on `maxage`, `method`, `onerror`, `request_body`, and `request_header`

Example

The following ESI output specifies `logindata` to refer to the environment variables stored in `catalog.xml`. `catalog.xml` enables access to the value of the `vendorID` environment variable, which is used as a parameter in the included URL:

```
<esi:environment src="/catalog.xml" name="logindata"/>
<esi:include
src="http://provider.com/intranetprovider?vendorID=$(logindata{vendorID})"/>
```

`catalog.xml` has the following content:

```
<?xml version=1.0?>
<esi-environment esiversion=""ORAESI/9.0.2"">
  <product_description>stereo</product_description>
  <vendorID>3278</vendorID>
  <partner1>E-Electronics</partner1>
  <partner2>E-City</partner2>
</esi-environment>
```

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.
- Oracle9*i*AS 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
- A null value evaluates whether or not a variable is empty

  When a number is compared with `null`, that number is converted into an equivalent string and compared against an empty string. In the following ESI markup, `$(logindata{name})` is a variable that provides access to the value of the `name`. If `name` is empty, then the expression evaluates to true; if `name` is not empty, then the expression evaluates to false.

  ```
  <esi:choose>
    <esi:when test="$(logindata{name}) == null">
      <esi:include src=/login/$(logindata{name})"/>
   </esi:when>
   <esi:otherwise>
     <esi:include src=/login/guest.html"/>
   <esi:otherwise>
  </esi:choose>
  ```

The following expressions show correct usage of booleans:

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

The following expressions show incorrect usage of booleans:

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

Statements

Statements must be placed inside a <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'">
    <img src="/img/TheImage.gif">
  </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

Oracle9*i*AS Web Cache first processes the contents of <esi:attempt>.

A failed <esi:attempt> triggers an error and causes Oracle9*i*AS Web Cache to process the contents of the <esi:except> tag.

Example

The following ESI markup attempts to fetch an advertisement. If the advertisement cannot be included, Oracle9*i*AS 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 Oracle9*i*AS Web Cache. If comments need to be visible in the HTML output, then 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://wwww.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 ouput*</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 Oracle9*i*AS 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 <A HREF=...> 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, Oracle9*i*AS Web Cache removes the starting `<!--esi` and ending `-->` 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 or custom environment variable outside of an ESI tag.

Syntax

<esi:vars>$(*environment_name*{*variable_name*})</esi:vars>

Usage

> **See Also:** "Variable Expressions" on page D-5 and "ESI inline Tag" on page D-19 for variable usage of ESI and custom variables

Example

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

```
<esi:vars>
  <IMG SRC="http://www.example.com/$(HTTP_COOKIE{type})/hello.gif"/ >
</esi:vars>
```

The following ESI output refers to `logindata` as part of the <A HREF=...> link for the Welcome page. `logindata` refers to an XML file that contains custom environment variables. The output also includes the user's `sessionID` and category `type` cookie values as part of the other <A HREF=...> links.

```
<esi:vars>
  <A HREF="welcome.jsp?name=$(logindata{name})">
  <A HREF="/shopping.jsp?sessionID=$(QUERY_STRING{sessionID})&type=$(QUERY_
STRING{type})">
  <IMG SRC="/img/shopping.gif">
  </A>
  <A HREF="/news.jsp?sessionID=$(QUERY_STRING{sessionID})&type=$(QUERY_
STRING{type})">
  <IMG SRC="/img/news.gif">
  </A>
  <A HREF="/sports.jsp?sessionID=$(QUERY_STRING{sessionID})&type=$(QUERY_
STRING{type})">
  <IMG SRC="/img/sports.gif">
  </A>
  <A HREF="/fun.jsp?sessionID=$(QUERY_STRING{sessionID})&type=$(QUERY_
STRING{type})">
  <IMG SRC="/img/fun.gif">
```

```
  </A>
  <A HREF="/about.jsp?sessionID=$(QUERY_STRING{sessionID})&type=$(QUERY_
STRING{type})">
  <IMG SRC="/img/about.gif">
  </A>
 </esi:vars>
```

E

Event Log Messages

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

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 port port address ip_address`	Oracle9*i*AS Web Cache is accepting administration requests on the specified listening port number and IP address.
`Listening on INVALIDATION port port address ip_address`	Oracle9*i*AS Web Cache is accepting invalidation requests on the specified listening port number and IP address.
`Listening on NORM port port address ip_address`	Oracle9*i*AS Web Cache is accepting Web browser requests on the specified listening port number and IP address.
`Listening on STATISTICS port port address ip_address`	Oracle9*i*AS Web Cache is accepting statistics monitoring requests on the specified listening port number and IP address.
`The cache server is started by the admin server at startup`	The Oracle9*i*AS Web Cache **admin server process** started the `cache` server process.
`The admin server started successfully`	The Oracle9*i*AS Web Cache `admin` server process successfully started.
`Auto-Restart: WXE-00800 Auto-restart started successfully`	The Oracle9*i*AS Web Cache **auto-restart process** successfully started
`The cache server started successfully`	The Oracle9*i*AS Web Cache **cache server process** successfully started.
HTTPS Startup Initialization Events	
`SSLInitialize: Origin Server Wallet file file does not exist`	Oracle9*i*AS Web Cache is unable to open the **wallet** file intended for HTTPS requests to **origin servers**. **See Also**: "Wallet Cannot Be Opened" on page 10-6
`SSLInitialize: Wallet file file does not exist`	Oracle9*i*AS Web Cache is unable to open the wallet file intended for HTTPS requests to Oracle9*i*AS Web Cache or its operations ports. **See Also**: "Wallet Cannot Be Opened" on page 10-6

Table E–1 (Cont.) Information Events

Message	Description
`SSLInitialize: Wallet Autologin file file does not exist - Wallet does not appear to be autologin wallet`	Oracle9*i*AS Web Cache is unable to open the wallet without a password. Oracle9*i*AS Web Cache requires auto-login of wallets. **See Also**: "Wallet Cannot Be Opened" on page 10-6
`SSLInitialize: Origin Server Wallet Autologin file file does not exist - Wallet does not appear to be autologin wallet`	
Shutdown Events	
`SIGTERM caught - program will shut down once all connections are complete.`	A UNIX event that specifies that Oracle9*i*AS Web Cache will shut down once all connections are complete.
`The server is exiting`	Oracle9*i*AS Web Cache is shutting down.
Operational Events	
`There was a network failure before the transaction was completed`	Oracle9*i*AS Web Cache terminated a connection to the Web browser.
Invalidation Events	
`<Invalidation>Exact URI` *`URI`* `has been invalidated successfully`	The URL is successfully invalidated.
`<Invalidation>Invalidation sent upstream to webcache host '`*`web_cache_hostname`*`'port '`*`port`*`' has returned with response code: '`*`response_code`*`'.`	In a **ESI cache hierarchy**, the **provider cache** is unable to connect to the **subscriber cache**.
`<Invalidation>Invalidation with info '`*`INFO_comment`*`' has returned with status '`*`status`*`'; number of documents invalidated: '`*`number`*`'`	The result of the invalidation. *`INFO_comment`* is the comment specified in the `INFO` element of the invalidation request. *`status`* is one of the following: ■ `SUCCESS` for successful invalidations ■ `URI NOT CACHEABLE` for documents that are not cacheable ■ `URI NOT FOUND` for documents not found *`number`* is the number of documents invalidated during the invalidation request.
`<Invalidation>Requested URI` *`URI`* `is not found in the cache. URI is not invalidated`	The URL is not in the cache and cannot be invalidated.

Table E–1 (Cont.) Information Events

Message	Description
<Invalidation>URI *URI* is not cacheable	The URL is not a cacheable document and cannot be invalidated.
<Invalidation>*number* URLs with prefix *prefix* have been successfully invalidated	The number of URLs by a particular prefix that have been successfully invalidated.
<Invalidation>Subscriber host '*web_cache_hostname*' port '*port*' has been removed due to exceeded failure count.	In a hierarchy of Oracle9*i*AS Web Cache servers, the invalidation provider's connection to the invalidation subscriber has failed more than three times.
Cache Cluster Event	
A *number* node cluster successfully initialized	Oracle9*i*AS Web Cache started the **cache cluster** with the specified number of cache cluster members.

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.
`Auto-Restart:WXE-08505` *`number`* `consecutive error(s) pinging the cache server.` *`number`* `error(s) required for restart.`	The auto-restart process encountered an error when polling the `cache` server process,
HTTPS Startup Initialization Events	
`Origin Server Wallet Failed to open at location location.`	Oracle9*i*AS Web Cache is unable to open the wallet intended for HTTPS requests to the origin server. **See Also:** "Wallet Cannot Be Opened" on page 10-6
`Warning: SSLInitialize: Orgin Server Wallet did not open -- Operating without wallet for backend`	Oracle9*i*AS Web Cache will use HTTPS for requests to the origin server without a wallet. **See Also:** "Wallet Cannot Be Opened" on page 10-6
`Wallet Failed to open at location location -- Opened wallet as user=user. -- Please verify wallet location and Auto Login support enabled.`	Oracle9*i*AS Web Cache found the wallet, but is unable to open the wallet without a password. Oracle9*i*AS Web Cache requires auto-login of wallets. **See Also:** "Wallet Cannot Be Opened" on page 10-6
Memory-Related Events	
`No space left for adding the Content-Length header for KeepAlive headers URI:` *`URI`*	Oracle9*i*AS 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, Oracle9*i*AS Web Cache allocates extra memory for `Keep-Alive` headers.
`Response cookie header too large`	The response cookie from the origin server is too large for Oracle9*i*AS Web Cache.

Table E–2 (Cont.) Warning Events

Message	Description
Origin Server Events	
`<Admin Server>Concurrent administration exceeded limit`	The number of concurrent connections (capacity) to the origin servers has been exceeded
`Connect Failed: Origin Web Server not accepting Connection`	The origin server is not accepting connections from Oracle9*i*AS Web Cache. This event could indicate that the origin server is down.
`Last-Modified time` *`time`* `is AFTER current time` *`time`*`, using current time instead`	The `Last-Modified` field in the response header is after the current time. Oracle9*i*AS Web Cache will use the current time instead.

Table E–2 (Cont.) Warning Events

Message	Description

`Origin Server module got an ABORT; Errno: error_number URI: URI`	Oracle9*i*AS Web Cache detects a problem with the origin server. *`error_number`* can be one of the following:
	`3 - low memory`
	There is not enough system memory for the `webcached` executable to run.
	`4 - LB fail`
	All of the origin servers are down, disabling the load balancing feature.
	`5 - no application server mapping. dynamic os not allowed`
	The error appears in either of the following situations
	—The site is not defined.
	—This virtual host site is not mapped to any origin servers.
	To create a site definition, use the Site Definitions page (**General Configuration** > **Sites**) of Oracle9*i*AS Web Cache Manager. To map a site to application Web servers or proxy servers, use the Site to Server Mapping page (**General Configuration** > **Site to Server Mapping**) of Oracle9*i*AS Web Cache Manager.
	`6 - socket connect fail`
	The origin server is down.
	`7 - socket create fail`
	The Oracle9*i*AS Web Cache computer has run short of system resources.
	`8 - send request fail`
	The request to the origin server has failed.
	`9 - bad input stream to send`
	Oracle9*i*AS Web Cache is unable to send the HTTP request body to the origin server.
	`10 - recv fail`
	Oracle9*i*AS Web Cache is unable to receive responses from the origin server.
	`11 - create header fail`
	Oracle9*i*AS Web Cache is unable to process the HTTP response header message from the origin server.

Table E–2 (Cont.) Warning Events

Message	Description

	`12 - header too big`
	The HTTP response header message from the origin server is too large.
	`13 - ct buffer too small`
	There is not enough system memory.
	`14 - ssl hand shake fail`
	The SSL hand shake between Oracle9*i*AS Web Cache and the origin server failed.
	`16 - ESI https mismatch`
	The protocol supported by origin server does not match the protocol specified in the `src` attribute of the `<esi:include>` tag.
	For example, if the `src` is `http://www.company.com/frag1` and `www.company.com` is a defined HTTPS origin server, then this message appears. You must change the `src` attribute to `https://www.company.com/frag1`.
	`17 - create request fail`
	Oracle9*i*AS Web Cache failed to create a request to the origin server.
	`18 - dynamic OS DNS fail`
	The DNS lookup for the ESI provider site failed. This can occur for the following reasons:
	—There is system memory problem with the Oracle9*i*AS Web Cache computer
	—The **ESI provider site** is down or does not exist
	`19 - dynamic OS connect fail`
	Oracle9*i*AS Web Cache received the DNS lookup result, but was unable to connect to the ESI provider site.
	`20 - header operation fail`
	Oracle9*i*AS Web Cache cannot manipulate the HTTP response header from the origin server. This occurs if the Oracle9*i*AS Web Cache computer is experiencing a system memory problem.

Table E–2 (Cont.) Warning Events

Message	Description

	`21 - OS request dropped`
	Oracle9*i*AS Web Cache cannot forward requests to the origin server. This occurs if the origin server has reached its capacity and the queue for new request is full.
	`22 - proxy authentication failed.`
	The authentication to the proxy server failed.
	`23 - proxy authentication failed for CONNECT`
	The proxy server denied the request.
	`24 - network error sending CONNECT to proxy`
	Oracle9*i*AS Web Cache encountered a network error sending a `CONNECT` message for opening an SSL connection to the proxy server. A `CONNECT` message open an SSL connection through the proxy.
	`25 - network error receiving response to CONNECT`
	Oracle9*i*AS Web Cache encountered a network error receiving a `CONNECT` message to the proxy server.
	`26 - proxy sent unrecognized response to CONNECT`
	Oracle9*i*AS Web Cache did not recognize the `CONNECT` response message sent by the proxy server.

Expiration time beyond year 2038, setting to MAX_LONG *time* time overflow, setting to MAX_LONG	The time set for the document goes beyond the year 2038, which is the maximum time limit.
Cache Cluster Events	
Add Cluster member# *cluster_ID*	Oracle9*i*AS Web Cache added the cache cluster member to the cache cluster.
Cluster Member sent invalid configuration error	A bad configuration response header was returned from a request. However, this request was not a Oracle9*i*AS Web Cache cluster peer request.
Cluster member# *cluster_ID* already marked alive	An attempt to mark the cache cluster member as active and running has been made. However, a previous concurrent request has already noted the member as being active and running.

Table E–2 (Cont.) Warning Events

Message	Description
`Cluster member# cluster_ID already marked dead`	An attempt to mark the cache cluster member as being inactive or down has been made. However, a previous concurrent request has already noted the member as being inactive or down.
`Cluster Member cluster_ID down. Start pinging.`	Oracle9*i*AS Web Cache detects a cache cluster member is down, and it starts to poll the member by sending requests a configured URL.
`Cluster Member cluster_ID up. Stop ping.`	Oracle9*i*AS Web Cache received a success response from the downed cache cluster member. Oracle9*i*AS Web Cache considers the cache cluster member to up again. It recalculates the relative capacity of the cache cluster members, and it reassigns ownership of cache content.
`Cluster Member cluster_ID - cluster_name configuration is invalid.`	The cache cluster member sent a message to Oracle9*i*AS Web Cache that indicates it has a configuration file different than the local configuration file for the cache cluster. A cluster runtime operates with the same `webcache.xml` configuration file for all members. To resolve this error, compare the two configuration files, and then propagate the proper version to all cluster members.
`Cluster Member cluster_ID - cluster_name configuration is valid.`	The cache cluster member was previously excluded from the cluster due to a configuration file that was different than the local view. The configuration files between these two members are now consistent.
`Cluster Member cluster_ID - cluster_name sent invalid configuration error.`	The cache cluster member sent a response to Oracle9*i*AS Web Cache that indicates it has a configuration file different than the local configuration file for the cache cluster. A cluster runtime operates with the same `webcache.xml` configuration file for all members. To resolve this error, compare the two configuration files, and then propagate the proper version to all cluster members.
`Remove Cluster member# cluster_ID`	Oracle9*i*AS Web Cache removed the cache cluster member from the cache cluster.
`Unknown Cluster Member sent invalid configuration error.`	The cache cluster member sent a response to Oracle9*i*AS Web Cache that indicates it has a configuration file different than the local configuration file for the cache cluster. Additionally, this peer member cache is not specified in the local `webcache.xml` configuration file. A cluster runtime operates with the same `webcache.xml` configuration file for all members. To resolve this error, compare the two configuration files, and then propagate the proper version to all cluster members.

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`	Oracle9*i*AS Web Cache is unable to open listening ports.
`Oracle Web Cache cache failed to initialize`	Oracle9*i*AS Web Cache is unable to initialize the `cache` server process.
`The server could not initialize`	Oracle9*i*AS Web Cache is unable to start.
`The server could not start service thread`	Oracle9*i*AS Web Cache encountered a thread initialization creation error.
`An error occurred scanning the directory` *`directory`*`.`	Oracle9*i*AS Web Cache is unable to load the Oracle9*i*AS Web Cache Manager help files or icons.
`Could not increase number of file/socket descriptors to` *`connections`*	Oracle9*i*AS Web Cache is unable to support the number connections. Adjust the number of maximum incoming in the Resource Limit page (**Cache-Specific Configuration** > **Resource Limits**) of Oracle9*i*AS Web Cache Manager. **See Also:** "Greater Than One Thousand Maximum Connections" on page 10-6
Startup Initialization Events Related to Auto-Restart Process	
`Auto-Restart:WXE-08500 Cache server not responding on port` *`port`*`.`	The Oracle9*i*AS Web Cache server may not be running.
`Auto-Restart:WXE-08501 Timeout occurred communicating with the cache server.`	Oracle9*i*AS Web Cache is either in an unstable state or slow responding to queries.
`Auto-Restart:WXE-08502 Lost connection with the cache server`	There were communication errors between the `auto-restart` process and the `cache` server process.
`Auto-Restart:WXE-08503 Operating system level error in` *`function error_number`*`:` *`error_message`*`.`	The `auto-restart` process encountered a system call returned error.

Table E–3 (Cont.) Error Events

Message	Description
`Auto-Restart:08504 number consecutive errors pinging the cache server.`	The number of consecutive failed requests to the `cache` server process that the `auto-restart` process has sent. Set the number of failed consecutive request in the Auto-Restart page (**General Configuration > Auto-Restart**) of Oracle9*i*AS Web Cache Manager. **See Also:** "Task 6: Configure Auto-Restart Process Settings" on page 6-12
`Auto-Restart:WXE-08506 Unable to read cache server pid file. Stopping the cache monitor.`	The `auto-restart` process is unable to read the `webcache.pid` file to determine the process ID of the `cache` server process. Most likely the file was deleted during a normal termination of the server.
`Auto-Restart:WXE-08507 Cache server process not running.`	There is no process on the system matching the process ID listed in the `webcache.pid` file.
`Auto-Restart:WXE-08508 Restarting the cache server.`	The `auto-restart` process is restarting the `cache` server process.
`Auto-Restart:WXE-08509 Starting the cache server.`	The `auto-restart` process is starting the `cache` server process.
`Failed to start the auto-restart monitor. failure_reson.`	The `cache` server process is unable to start the `auto-restart` process.
Read/Write Events	
`Could not open config file (config_file)`	Oracle9*i*AS Web Cache is unable to write to its configuration file due to a permissions problem.
`Could not open log file (access_log)`	Oracle9*i*AS Web Cache is unable to write to its access log file due to a permissions problem.
UNIX Process Identity Events	
`Failed to find User (user) in /etc/password` `Invalid User ID (user)`	The current user cannot perform this operation. Only the root user or the user specified in the Process Identity page (**Cache-Specific Configuration** > **Process Identity**) of the Oracle9*i*AS Web Cache Manager can perform this operation.

Table E–3 (Cont.) Error Events

Message	Description
`Failed to find Group (`*`group`*`) in /etc/group` `Invalid Group ID (`*`group`*`)`	The group ID that the current user is a member of is not valid for this operation.
`Permission denied when setting User ID (`*`user`*`)`	The current user that is being set is not the owner of the Oracle9*i*AS Web Cache executables and is not the root user.
`Permission denied when setting Group ID (`*`group`*`)`	The current group that is being set is not the owner of the Oracle9*i*AS Web Cache executables.
Memory-Related Events	
`Cache failed to allocate memory for the hash table`	Oracle9*i*AS Web Cache is unable to allocate memory for cache initialization.
`Cache Index memory allocation error`	Oracle9*i*AS Web Cache is unable to allocate memory for cache initialization.
`Document compression error: `*`error`*	Oracle9*i*AS Web Cache is unable to compress the document in its cache.
`Insert failed (memory low during insertion) for slave document `*`document`*	Oracle9*i*AS Web Cache is unable to insert a document into its cache.
`Out of memory`	Oracle9*i*AS Web Cache is out of cache memory. Adjust the cache memory in the Resource Limits page (**Cache-Specific Configuration** > **Resource Limits**) of Oracle9*i*AS Web Cache Manager.
`Process out of memory on malloc/realloc request. Exiting process.`	Oracle9*i*AS Web Cache is out of cache memory. Adjust the cache memory in the Resource Limits page (**Cache-Specific Configuration** > **Resource Limits**) of Oracle9*i*AS Web Cache Manager.

Table E–3 (Cont.) Error Events

Message	Description
`The system has run critically low on memory`	This error indicates that insufficient virtual memory remains in the `cache` server process to satisfy the request. The error can be caused by either of the following situations: ■ The request is for an inordinate amount of memory, thus causing the system limit for virtual memory to be exceeded. ■ The cache's memory is extremely full and this request puts the `cache` server process over the system limit. When this error occurs, the `cache` server process is stopped. If auto-restart is enabled, then the `auto-restart` process automatically restarts the `cache` server process. If auto-restart is not enabled, then restart the `cache` server process from the Operations page (**Administration** > **Operations**).
Operational Events	
`Invalid XLF Field Name: user_field`	The user-specified field specified for the access log file is invalid. The fields are specified in the Access Logs page (**Cache-Specific Configuration** > **Access Logs**) of Oracle9*i*AS Web Cache Manager.
`Too many session definitions`	The number of allowed session definitions used for cacheability rules for pages with personalized attributes or session tracking has exceed the 20 name limit. Reduce the name of session names in Session / Personalized Attribute Definitions (**General Configuration** > **Session Management > Session/Personalized Attribute Definitions**) page of Oracle9*i*AS Web Cache Manager.
Origin Server Events	
`Unable to resolve the IP address of ip_address. Check your DNS setup.`	Oracle9*i*AS Web Cache is unable to resolve the IP address of the origin server. You can alter origin server configuration in the Application Web Servers page (**General Configuration** > **Application Web Servers**) or the Proxy Server page (**General Configuration** > **Proxy Servers**) of Oracle9*i*AS Web Cache Manager.

Table E–3 (Cont.) Error Events

Message	Description
`This product only supports IPv4 for origin server ip_address. Check your DNS setup`	Oracle9*i*AS Web Cache supports IP version 4. The IP address of the origin server cannot be resolved because it uses another version of the IP.
Invalidation Events	
`<Invalidation>Cannot compose key pattern for the requested URI URI`	The URL specified in the invalidation message does not have a corresponding cacheability rule.
`<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 (**General Configuration** > **Security**) of Oracle9*i*AS Web Cache Manager.
`<Invalidation>Default URL size too small for cache key`	The URL specified in the invalidation message is too long. Oracle9*i*AS Web Cache has a 3 KB limit on URLs that may or may not include cookies or HTTP request headers.
`<Invalidation>Empty entity`	The invalidation message is empty.
`<Invalidation>Invalid validity level (valid range is 0-9). Level=level.`	The validity level specified in the invalidation message is not valid.
`<Invalidation>Not an invalidation request`	The message is not an invalidation message.
`<Invalidation>Unrecognized cookies in the invalidation message`	The cookies specified in the invalidation message are not valid.
`<Invalidation>URL Node reading error`	The URL specified in the invalidation message is invalid or or there is a memory allocation problem.
`<Invalidation>Username/password check failed. Access denied.`	The invalidation user name and password is not valid. The invalidation user is invalidator. By default, the password is `invalidator`. You can change the password in the Security page (**General Configuration** > **Security**) of the Oracle9*i*AS Web Cache Manager.
`<Invalidation>XML parsing error`	The invalidation message uses invalid XML syntax.

Table E–3 (Cont.) Error Events

Message	Description
Cache Cluster Events	
`Cache memory allocation for the cluster configuration block`	Oracle9*i*AS Web Cache ran out of memory on the system.
`Cache memory allocation for the cluster member table`	Oracle9*i*AS Web Cache ran out of memory on the system.
`Cluster member count` *`count`* `is larger than maximum allowed 99`	The number of cache cluster members specified in the cache cluster configuration is larger than the maximum limit of 99.

F

Using Oracle9*i*AS Web Cache with Third-Party Application Web Servers

This chapter discusses how to configure Oracle9*i*AS Web Cache with third-party application Web servers.

This chapter contains these topics:

Notes:

- While this chapter describes how Oracle9*i*AS Web Cache works with three specific kinds of servers, Oracle9*i*AS Web Cache works with any HTTP-compliant application Web server.
- The application examples used in the discussions of these third-party servers are relatively simple. Running with production applications will require more extensive configuration of Oracle9*i*AS Web Cache. Refer to the third-party application Web server documentation for information about designing applications.

Overview of Third-Party Application Servers

Because Oracle9*i*AS Web Cache is transparent to the application Web server, the application Web server treats HTTP requests from Oracle9*i*AS Web Cache as any other HTTP request coming directly from the browser. In turn, the application Web server generates the response and sends it back to Oracle9*i*AS Web Cache as an HTTP message.

Because Oracle9*i*AS Web Cache fully supports HTTP, it can work with any HTTP-compliant application Web server. How the application Web servers choose to generate HTTP responses is irrelevant to Oracle9*i*AS Web Cache.

The type of application Web server that a site uses depends mainly on the types of applications that site is running. For example, if customers want to run Active Server Pages (ASP), then they may prefer to use Microsoft Internet Information Server (IIS) as the application Web server.

This section contains these topics:

Web-Site Configuration

You configure Oracle9*i*AS Web Cache to communicate with a third-party application Web servers the same way you would with Oracle HTTP Server, by providing the host name and the listening port number. The default values for the listening ports for the products discussed in this chapter are given in Table F–1.

Table F–1 Third-Party Application Web Server Default Listening Ports

Application Web Server	Port
BEA WebLogic Server 6.0	7001
IBM WebSphere Application Server, Version 4.0	80
Microsoft IIS 5.0	80

To configure Oracle9*i*AS Web Cache to communicate with a third-party application Web server, perform the following tasks:

- "Task 9: Specify Settings for Origin Servers" on page 6-23 to configure application Web server settings
- "Task 10: Configure Web Site Settings" on page 6-26 to configure Web site settings
- "Task 3: Configure Oracle9iAS Web Cache with Listening Ports for Incoming Browser Requests" on page 6-6 to change Oracle9*i*AS Web Cache port settings

Caching Rules and Expiration Rules

You assign caching rules and expiration rules when using third-party application Web servers in the same way as when using Oracle HTTP Server. You can select to cache or not to cache content for the following:

- Static documents
- Multiple-version documents for the same URL
- Pages supporting a **session cookie** or **embedded URL parameter**
- Pages containing simple personalization
- Dynamic assembly of **Edge Side Includes (ESI)** fragments

You can also assign an expiration time limit to documents or invalidate documents at any time.

See Also: Chapter 7, "Creating Caching Rules"

BEA WebLogic Server 6.0

The WebLogic Server 6.0 installation includes a number of Java Server Pages (JSP), Java servlets, and Enterprise JavaBeans (EJB) examples. For the purposes of this section, the following examples are used:

- WebLogic SnoopServlet
- WebLogic SessionServlet

WebLogic SnoopServlet

The `SnoopServlet` servlet obtains and uses request information, headers, and parameters sent by the browser. Use it to demonstrate how Oracle9*i*AS Web Cache caches full-page dynamic content with `SnoopServlet`.

To start, perform the following:

1. Ensure that Oracle9*i*AS Web Cache has been configured to communicate with the WebLogic Application Server, as described in "Web-Site Configuration" on page F-3.
2. Start the WebLogic Server, and then access the following URL:

   `http://`*`hostname`*`:7001/examplesWebApp/SnoopServlet`

   When you access the URL, notice that your browser displays request information, headers, parameters, and the GIF image "Build On bea."

To cache the content:

1. Create a caching rule for the `SnoopServlet` output, as described in "Configuring Caching Rules" on page 7-8.

   When creating the caching rule for the `SnoopServlet` output, configure the following in the Edit Cacheability Rule dialog box:

   a. In the **URL Expression** field, enter `/examplesWebApp/SnoopServlet`.
   b. In the **Method** row, choose **GET**.
   c. Select **Cache**.
   d. Leave all other defaults in the Edit Cacheability Rule dialog box as is.

2. Point the browser to the Oracle9*i*AS Web Cache with following URL:

   `http://web_cache_hostname:7777/examplesWebApp/SnoopServlet`

   The output is the same as it was when you accessed `SnoopServlet` directly from the WebLogic Server. This time, Oracle9*i*AS Web Cache caches the `SnoopServlet` output and serves the request to the browser.

   Note: Port 7777 is the default listening port for Oracle9*i*AS Web Cache. If you changed the default listening port, use that port number.

3. View the contents of the cache, as described in "Listing the Contents of the Cache" on page 8-49, to ensure that `SnoopServlet` is cached.

   From this point on, anytime a browser accesses `SnoopServlet`, the response will be served from Oracle9*i*AS Web Cache.

WebLogic SessionServlet

The `SessionServlet` servlet provides a simple example of an HTTP servlet that uses the `HttpSession` class to track the number of times that a browser has visited the servlet. Use it to demonstrate how Oracle9*i*AS Web Cache caches pages with **session-encoded URLs**.

To start, perform the following:

1. Ensure that Oracle9*i*AS Web Cache has been configured to communicate with the WebLogic Server, as described in "Web-Site Configuration" on page F-3.
2. Configure the browser not to accept cookies.

   This is required in order to use session-encoded URLs in this example. Finally, start the WebLogic Server and access the following URL:

   `http://hostname:7001/examplesWebApp/SessionServlet`

   Notice that the page displays how many times a browser has visited it. When you click the link labeled "here," notice that the session ID is encoded in the URL. Every time you refresh or reload the page, the counter increases by one.

To cache the content:

1. Create an expiration rule, as described in "Configuring Expiration Rules" on page 7-18.

   In the Create Expiration Rule dialog box, perform the following:

   a. In the **Expire** section of the Create Expiration Rule dialog box, select to expire the output 60 seconds after cache entry.

   b. In the **After Expiration** section of the Create Expiration Rule dialog box, select **Remove immediately**.

2. Create a session-related caching rule, as described in "Configuring Session-Related or Personalized Attributed-Related Caching Rules" on page 7-38.

   When configuring a session-related caching rule for `SessionServlet`, perform the following steps:

   a. In the Create Session / Personalized Attribute Definition dialog box:

      * In the Session / Attribute field, enter `BEASession`.
      * In the Cookie Name field, enter `JSESSIONID`.

        `JSESSIONID` is the default cookie name used by the WebLogic Server 6.0.

      * In the URL Parameter field, enter `jsessionid`.

   b. In the Add Session / Personalized Attribute Related Caching Rule dialog box:

      * From the **Please select a session/attribute** list, select `BEASession`.
      * Select **YES** for prompt 1.
      * Select **YES** for prompt 2.
      * Select **NO** for prompt 3.

   c. Create a new caching rule for `SessionServlet`, as described in "Configuring Caching Rules" on page 7-8.

      When creating the caching rule for the `SessionServlet` output, configure the following in the Edit Cacheability Rule dialog box:

      * In the **URL Expression** field, enter `/examplesWebApp/SessionServlet`.
      * In the **Method** row, choose **GET**.
      * Select **Cache**.
      * In the **Expiration Rule** row, select `Expire: 60 seconds in cache. After: remove immediately`.
      * In the Session / Personalized Attribute Related Caching Rules row, select **Apply the following** and `BEASession: cache with session, cache w/o session`.
      * Leave all other defaults in the Edit Cacheability Rule dialog box as is.

3. Point the browser to Oracle9*i*AS Web Cache with the following URL:

   `http://web_cache_hostname:7777/examplesWebApp/SessionServlet`

   The output is the same as it was when you accessed `SessionServlet` directly from the WebLogic Server. This time Oracle9*i*AS Web Cache caches the `SessionServlet` output. When the page is refreshed or reloaded, notice that the counter does not increment by one. This is because Oracle9*i*AS Web Cache serves the content, and the request never goes to the WebLogic Server.

   Note: Port 7777 is the default listening port for Oracle9*i*AS Web Cache. If you changed the default listening port, use that port number.

4. View the contents of the cache, as described in "Listing the Contents of the Cache" on page 8-49, to ensure that `SessionServlet` is cached.

   When you reload the page, notice that the cached response appears faster than when you access the WebLogic server directly.

   Because the expiration rule for this URL is set to 60 seconds, Oracle9*i*AS Web Cache expires the cached content after 60 seconds and reflects the content the next time the user requests the page.

IBM WebSphere Application Server, Version 4.0

The WebSphere Application Server, Version 4.0 installation includes a number of JSP, Java servlets, and EJB examples. For the purposes of this section, the following examples are used:

- WebSphere Snoop Servlet
- WebSphere SessionSample

WebSphere Snoop Servlet

The snoop servlet shows getting and using request information, headers, and parameters sent by the browser. Use it to demonstrate how Oracle9*i*AS Web Cache caches full-page dynamic content.

To start, perform the following:

1. Ensure that Oracle9*i*AS Web Cache has been configured to communicate with the WebSphere Application Server, as described in "Web-Site Configuration" on page F-3.
2. Start the WebSphere Application Server, and then access the following URL:

   `http://hostname/servlet/snoop`

   Notice that request information, headers, and parameters sent by your browser are displayed.

To cache this content:

1. Create a caching rule for the snoop output, as described in "Configuring Caching Rules" on page 7-8.

   When creating the caching rule for the snoop output, configure the following in the Edit Cacheability Rule dialog box:

   a. In the **URL Expression** field, enter `/servlet/snoop`.
   b. In the **Method** row, choose **GET**.
   c. Select **Cache**.
   d. Leave all other defaults in the Edit Cacheability Rule dialog box as is.

2. Point the browser to the Oracle9*i*AS Web Cache with following URL:

   `http://web_cache_hostname:7777/servlet/snoop`

   The output is the same as it was when you accessed `snoop` directly from the WebSphere Application Server. This time, Oracle9*i*AS Web Cache caches the `snoop` output and serves the response to the browser.

   Note: Port 7777 is the default listening port for Oracle9*i*AS Web Cache. If you changed the default listening port, use that port number.

3. View the contents of the cache, as described in "Listing the Contents of the Cache" on page 8-49, to ensure that `snoop` is cached.

   When you reload the page, you should notice that the cached response appears faster than when you access the WebSphere Application Server directly.

WebSphere SessionSample

The `SessionSample` servlet is a simple example of an HTTP servlet that tracks the number of times that a browser has visited the servlet using a cookie. Use it to demonstrate how Oracle9*i*AS Web Cache caches pages with session cookies.

This example is not a pre-deployed WebSphere example like the `snoop` servlet. You can find this example in Section 4.4.1.1: Session programming model and environment in the WebSphere Application Server online documentation, when you click the `SessionSample.java` link on that page.

To start, perform the following:

1. Compile the `SessionSample.java` file in the WebSphere environment.
2. Copy the `SessionSample.class` file in the location where the `snoop.class` file resides.

   The default location for the `snoop.class` file is WebSphere's install directory:

   `\WebSphere\AppServer\hosts\default_host\default_app\servlets\`

3. Start the WebSphere Application Server, set the browser to accept cookies, and then access the following URL:

   `http://`*`hostname`*`/servlet/SessionSample`

   Notice that the page displays the number of times a browser has visited this page. When you reload this page, the counter increments by one.

To cache this content:

1. Create an expiration rule, as described in "Configuring Expiration Rules" on page 7-18.

   In the Create Expiration Rule dialog box, perform the following:

   a. In the **Expire** section of the Create Expiration Rule dialog box, select to expire the output 60 seconds after cache entry.

   b. In the **After Expiration** section of the Create Expiration Rule dialog box, select **Remove immediately**.

2. Create a session-related caching rule, as described in "Configuring Session-Related or Personalized Attributed-Related Caching Rules" on page 7-38.

   When configuring a session-related caching rule, perform the following steps:

   a. In the Create Session / Personalized Attribute Definition dialog box:

      * In the Session / Attribute field, enter `IBMSession`.
      * In the Cookie Name field, enter `sessionid`.

        `sessionid` is the default cookie name used by the WebSphere Application Server, Version 4.0.

      * In the URL Parameter field, enter `sessionid`.

   b. In the Add Session / Personalized Attribute Related Caching Rule dialog box:

      * From the **Please select a session/attribute** list, select `IBMSession`.
      * Select **YES** for prompt 1.
      * Select **YES** for prompt 2.
      * Select **NO** for prompt 3.

   c. Create a new caching rule for `SessionSample`, as described in "Configuring Caching Rules" on page 7-8.

      When creating the caching rule for the `SessionSample` output, configure the following in the Edit Cacheability Rule dialog box:

      * In the **URL Expression** field, enter `/servlet/SessionSample`.
      * In the **Method** row, choose **GET**.
      * Select **Cache**.
      * In the **Expiration Rule** row, select `Expire: 60 seconds in cache. After: remove immediately`.
      * In the Session / Personalized Attribute Related Caching Rules row, select **Apply the following** and `IBMSession: cache with session, cache w/o session`.
      * Leave all other defaults in the Edit Cacheability Rule dialog box as is.

3. Point the browser to Oracle9*i*AS Web Cache with the following URL:

   `http://web_cache_hostname:7777/servlet/SessionSample`

   The output is the same as when you access `SessionSample` directly from WebSphere Application Server. This time, Oracle9*i*AS Web Cache caches the `SessionSample` output. To verify that the content is served by the cache, refresh or reload the page. Notice that the counter remains the same. This is because Oracle9*i*AS Web Cache serves the content, and the request never goes to WebSphere Application Server.

   Note: Port 7777 is the default listening port for Oracle9*i*AS Web Cache. If you changed the default listening port, use that port number.

4. View the contents of the cache, as described in "Listing the Contents of the Cache" on page 8-49, to ensure that `SessionSample` is cached.

   When you reload the page, notice that the cached response appears faster than when you access the WebSphere server directly.

   Because the expiration rule for this URL is set to 60 seconds, Oracle9*i*AS Web Cache expires the cached content after 60 seconds and reflects the content the next time the user requests the page.

Microsoft IIS 5.0

The IIS 5.0 installation includes a number of ASP examples. For the purposes of this section, the following examples are used:

- ServerVariables_Jscript ASP
- Cookie_Jscript ASP

ServerVariables_Jscript ASP

`ServerVariables_JScript.asp` demonstrates techniques you can use to access server variable information from an ASP script. Use it to demonstrate how Oracle9*i*AS Web Cache caches full-page dynamic content.

To start, perform the following:

1. Ensure that Oracle9*i*AS Web Cache has been configured to communicate with IIS, as described "Web-Site Configuration" on page F-3.
2. Start IIS, and then access the following URL:

   `http://`*`hostname`*`/IISSamples/sdk/asp/interaction/ServerVariables_JScript.asp`

   Notice that request information, headers, and parameters sent by the browser are displayed.

To cache this content:

1. Create a caching rule for the `ServerVariables_JScript.asp`, as described in "Configuring Caching Rules" on page 7-8.
   a. In the **URL Expression** field, enter `/IISSamples/sdk/asp/interaction/ServerVariables_JScript.asp`.
   b. In the **Method** row, choose **GET**.
   c. Select **Cache**.
   d. Leave all other defaults in the Edit Cacheability Rule dialog box as is.

2. Point the browser to the Oracle9*i*AS Web Cache with following URL:

   `http://`*`web_cache_hostname`*`:7777/IISSamples/sdk/asp/interaction/ServerVariables_JScript.asp`

   The output is the same as it was when you accessed `ServerVariables_JScript.asp` directly from IIS. This time, Oracle9*i*AS Web Cache caches the `ServerVariables_JScript.asp` output and serves the request to the browser.

   Note: Port 7777 is the default listening port for Oracle9*i*AS Web Cache. If you changed the default listening port, use that port number.

3. View the contents of the cache, as described in "Listing the Contents of the Cache" on page 8-49, to ensure that `ServerVariables_JScript.asp` is cached.

   When you reload the page, you should notice that the cached response appears faster than when you access IIS directly.

Cookie_Jscript ASP

`Cookie_JScript.asp` illustrates how your script can set and read cookies by using the `Response.Cookies` collection. Use it to demonstrate how Oracle9*i*AS Web Cache caches pages with session cookies.

To start, perform the following:

1. Ensure that Oracle9*i*AS Web Cache has been configured to communicate with IIS, as described in "Web-Site Configuration" on page F-3.
2. Start IIS, verify that your browser is set to accept cookies, and then access the following URL:

   `http://`*`hostname`*`/IISSamples/sdk/asp/interaction/Cookie_JScript.asp`

   When you access the URL, notice that the page displays the date and time you last visited this page. When you click "Revisit this page," the date and time is updated.

To cache this content:

1. Create an expiration rule, as described in "Configuring Expiration Rules" on page 7-18.

   In the Create Expiration Rule dialog box, perform the following:

   a. In the **Expire** section of the Create Expiration Rule dialog box, select to expire the output 60 seconds after cache entry.

   b. In the **After Expiration** section of the Create Expiration Rule dialog box, select **Remove immediately**.

2. Create a session-related caching rule for `Cookie_Jscript.asp`, as described in "Configuring Session-Related or Personalized Attributed-Related Caching Rules" on page 7-38.

   When configuring a session-related caching rule, perform the following steps:

   a. In the Session / Attribute field, enter `MSSession`.

   b. In the Cookie Name field, enter `CookieJSCript`.

c. In the Add Session / Personalized Attribute Related Caching Rule dialog box:

* From the **Please select a session/attribute** list, select `MSSession`.
* Select **YES** for prompt 1.
* Select **YES** for prompt 2.
* Select **NO** for prompt 3.

d. Create a new caching rule for `Cookie_JScript.asp`, as described in "Configuring Caching Rules" on page 7-8.

e. When creating the caching rule for the `Cookie_JScript.asp` output, configure the following in the Edit Cacheability Rule dialog box:

* In the **URL Expression** field, enter `/IISSamples/sdk/asp/interaction/Cookie_JScript.asp`.
* In the **Method** row, choose **GET**.
* Select **Cache**.
* In the **Expiration Rule** row, select `Expire: 60 seconds in cache. After: remove immediately`.
* In the Session / Personalized Attribute Related Caching Rules row, select **Apply the following** and `MSSSession: cache with session, cache w/o session`.
* Leave all other defaults in the Edit Cacheability Rule dialog box as is.

3. Point the browser to Oracle9*i*AS Web Cache with the following URL:

`http://`*`web_cache_hostname`*`:7777/IISSamples/sdk/asp/interaction/Cookie_JScript.asp`

The output is the same as it was when you accessed `Cookie_JScript.asp` directly from IIS. This time, Oracle9*i*AS Web Cache caches the `Cookie_JScript.asp` output. To verify that the cache serves the content, click "Revisit this page." Notice that the date and time are not updated. This is because Oracle9*i*AS Web Cache serves the cached content, and the request never goes to IIS.

Note: Port 7777 is the default listening port for Oracle9*i*AS Web Cache. If you changed the default listening port, use that port number.

---

4. View the contents of the cache, as described in "Listing the Contents of the Cache" on page 8-49, to ensure that `Cookie_JScript.asp` is cached.

   When you reload the page, notice that the cached response appears faster than when you access IIS server directly.

   Because the expiration rule for this URL is set to 60 seconds, Oracle9*i*AS Web Cache expires the cached content after 60 seconds and reflects the content the next time the user requests the page.

Glossary

access log

A log file that contains information about the HTTP requests sent to Oracle9*i*AS 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 Oracle9*i*AS Web Cache process that provides administration, configuration, and monitoring capabilities.

application Web server

An **origin 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.

auto-restart process

An Oracle9*i*AS Web Cache process that checks if the **cache server process** server process is running and automatically restarts the `cache` server process if it is not running

cache hit

An HTTP Web browser request that can be satisfied from the Oracle9*i*AS Web Cache cache without going to the **origin server**.

cache cluster

A loosely coupled collection of cooperating Oracle9*i*AS Web Cache cache instances working together to provide a single logical cache. Cache clusters provide failure detection and failover of Web caches, increasing the availability of your Web site.

cache cluster member

An instance of Oracle9*i*AS Web Cache configured with other instances of Oracle9*i*AS Web Cache to operate as one logical cache. The cache cluster members communicate with one another to request cacheable content that is cached by another cache cluster member and to detect when a cache cluster member fails.

cache hierarchy

A deployment in which an Oracle9*i*AS Web Cache caches content from another Oracle9*i*AS Web Cache to a local market. Oracle9*i*AS Web Cache provides support for a **distributed cache hierarchy** in a distributed network and an **ESI cache hierarchy** in an **ESI provider site** configuration.

cache miss

An HTTP Web browser request that cannot be satisfied from the Oracle9*i*AS Web Cache cache and most go to an **origin server**.

cache server process

An Oracle9*i*AS Web Cache process that manages the cache by providing connection management and request processing.

capacity

The maximum number of concurrent connections that the **origin server** can accept.

category cookie

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

central cache

In a **distributed cache hierarchy**, an Oracle9*i*AS Web Cache server that acts as an **origin server** to at least one **remote cache**. When content becomes invalid, the central cache propagates the invalidation request to the remote caches to ensure consistency.

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 **origin server** to a Web browser during an HTTP request. During subsequent HTTP requests, the cookie is passed back to the origin server, enabling the origin 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

distributed cache hierarchy

A **cache hierarchy** in which a **central cache** acts as an **origin server** to a **remote cache**.

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. Oracle9*i*AS Web Cache accepts requests that use the following characters as delimiters: question mark (?), ampersand (&), dollar sign ($), or semi-colon (;).

ESI

See **Edge Side Includes (ESI)**.

ESI cache hierarchy

A **cache hierarchy** in which a **provider cache** acts as an **origin server** to a **subscriber cache**.

ESI provider site

A site that Oracle9*i*AS Web Cache contacts for **Edge Side Includes (ESI)** assembly only. Browsers are not allowed to request content from these sites.

event log

A log file that contains Oracle9*i*AS 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.

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 **origin server** fails, Oracle9*i*AS Web Cache automatically distributes the load over the remaining origin servers and polls the failed origin server for its current up / down status until it is back online. In a cache cluster environment, Oracle9*i*AS Web Cache transfers ownership of the content of the failing member to the remaining cluster members.

failure detection

In a cache cluster environment, Oracle9*i*AS Web Cache detects when a cache cluster member is unavailable.

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 **origin server** 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 the **origin server** to communicate.

HTTP request header

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

HTTP request method

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

HTTPS protocol

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

invalidation

Oracle9*i*AS Web Cache function that marks documents as invalid and then refreshes them with updated content from the **origin server**. Invalidation keeps the Oracle9*i*AS Web Cache cache consistent with the content on the origin servers.

invalidation coordinator

In a cache cluster environment, Oracle9*i*AS Web Cache propagates invalidation messages to other cache cluster members. It sends the invalidation messages to one cache cluster member who acts as the coordinator**.** The coordinator propagates the invalidation messages to the other cluster members.

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 **origin server**s 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.

Load Balancer

A network switch that balances the load of incoming browser request. In an Oracle9*i*AS Web Cache deployment, the Load Balancer is typically positioned in front of the Oracle9*i*AS Web Cache server.

on-demand content

In a cache cluster environment, on-demand content are popular documents that are stored in the cache of each cluster member.

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 following it and provides a service to the preceding layer.

OSI

See **Open Systems Interconnection (OSI)**.

Oracle Enterprise Manager

A tool for administering Oracle9*i* Application Server. It is a complete management solution for administering, configuring, and monitoring the application server and its components. Using it, you can:

- View the overall status of Oracle9*i*AS Web Cache
- View performance metrics

Oracle9*i*AS Web Cache Manager

A tool that combines configuration abilities with component control to provide an integrated environment for configuring and managing Oracle9*i*AS Web Cache.

origin server

A server that is either an **application Web server** for internal sites or a **proxy server** for external sites protected by a firewall.

owned content

In a cache cluster environment, content that is owned by a particular cache cluster member. Oracle9*i*AS Web Cache distributes the cached content among the cache

cluster members. In effect, it assigns content to be owned by a particular cache cluster member.

partial page caching

A feature that enables Oracle9*i*AS 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 Oracle9*i*AS 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 Oracle9*i*AS Web Cache to assign a queue order to documents. These heuristics determine which documents can be served stale and which documents must be retrieve 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 **origin server** is not exceeded, then the most popular and least valid documents are refreshed first.

personalized attribute

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 Oracle9*i*AS Web Cache to 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 **origin 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 origin server or database. You should only cache pages that use POST forms if they are used in searches.

---

proxy server

An **origin server** that substitutes for the real server, forwarding client connection requests to the real server or to other proxy servers. Proxy servers provide access control, data and system security, monitoring, and caching.

provider

Set of content—content areas, pages, applications, even data from outside sources—brought together in one central location and accessed through a common interface, called a page.

provider cache

In an **ESI cache hierarchy**, an Oracle9*i*AS Web Cache server that locally caches content for a **provider site**. A **subscriber cache** then contacts the provider caches for assembly of HTML fragments. When content becomes invalid, the provider cache propagates the invalidation request to the subscriber cache to ensure consistency.

provider site

A site that provides as a source of content for a **provider cache** and a **subscriber cache**.

regular expression

Oracle9*i*AS 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

remote cache

In a **distributed cache hierarchy**, an Oracle9*i*AS Web Cache server that caches content from a **central cache** to serve local requests. When an invalidation request is sent to the central cache, the central cache propagates the request to the remote cache, ensuring consist content.

reverse proxy server

A proxy server that appears to be a normal server to browsers but internally retrieves its documents from other **origin server**s 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, Oracle9*i*AS 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 **origin 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-encoded URLs

HTML hyperlink tags, such as `<A HREF=...>`, that contain embedded session information to distinguish users. You can configure Oracle9*i*AS Web Cache to substitute the values of session parameters in HTML hyperlink tags with the session information contained within a **session cookie** or an **embedded URL parameter**.

subscriber cache

In an **ESI cache hierarchy**, an Oracle9*i*AS Web Cache server that assembles ESI content by contacting a **provider cache** for the template's HTML fragments. The HTML fragments are then assembled. When provider site content becomes invalid, the provider site propagates the invalidation request to the **subscriber cache** to ensure consistency.

Uniform Resource Identifier (URI)

The address syntax that is used to create a **URL**.

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 Oracle9*i* 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.

Oracle9*i*AS 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.

virtual host site

A site hosted by Oracle9*i*AS Web Cache. Browsers can request cached content from these sites through Oracle9*i*AS Web Cache In addition to caching content, Oracle9*i*AS Web Cache can also assemble ESI fragments from these sites.

wallet

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.

weighted availability capacity

The percentage of the available **capacity** that the **origin server** can accept.

XML

See **Extensible Markup Language (XML)**.

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z