Oracle® Application Server Web Cache

Administrator's Guide

10*g* Release 2 (10.1.2)

B14046-05

February 2006

ORACLE®

Oracle Application Server Web Cache Administrator's Guide, 10*g* Release 2 (10.1.2)

B14046-05

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.

Contents

5 OracleAS Web Cache Topologies

6 Introduction to Administration Tools

7 Starting and Stopping OracleAS Web Cache

Part II   Configuration and Administration of OracleAS Web Cache

8 Setup and Configuration

9   Configuring OracleAS Web Cache for HTTPS Requests

10   Configuring Cache Clusters

11   Configuring a Hierarchy of Caches

12   Creating Caching Rules

13   Sending Invalidation Requests

14   Monitoring Cache Trends with Statistics

15   Using Diagnostics Tools

Part III   Reference

16   Edge Side Includes (ESI) Language Tags

17   Event Log Messages

A   OracleAS Web Cache Directory Structure

B   OracleAS Web Cache as a Standalone Product

E   Troubleshooting OracleAS Web Cache

Glossary

Index

Preface

Oracle Application Server Web Cache Administrator's Guide describes how to use Oracle Application Server Web Cache (OracleAS Web Cache) to cache both static and dynamically generated content for at least one **origin server**.

Audience

Oracle Application Server Web Cache Administrator's 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.

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. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

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

Accessibility of Code Examples in Documentation

Screen readers 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, some screen readers 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 does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documents

For more information, see:

- Online help available through Oracle Enterprise Manager 10*g* and Oracle Application Server Web Cache Manager

- The Oracle Application Server documentation set, especially:

    - *Oracle Application Server Web Cache Invalidation API Reference*

    - *Oracle Application Server Concepts*

    - *Oracle Application Server Administrator's Guide*

    - *Oracle Application Server Performance Guide*

    - *Oracle Application Server Security Guide*

    - *Oracle Application Server Containers for J2EE JSP Tag Libraries and Utilities Reference*

    - *Oracle Business Intelligence Discoverer Configuration Guide*

    - *Oracle Application Server Portal Configuration Guide*

    - *Oracle Application Server Single Sign-On Administrator's Guide*

    - *Oracle Application Server Wireless Administrator's Guide*

    - Oracle PL/SQL documentation

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in OracleAS Web Cache?

This section describes the new features of OracleAS Web Cache and provides pointers to additional information. This information is mostly useful to users who have managed Oracle Application Server 10*g* Release 2 (10.1.2.0.0).

The following sections describe the new features in OracleAS Web Cache:

- New Features in 10g Release 2 (10.1.2.0.2)
- New Features in 10g Release 2 (10.1.2.0.0)

New Features in 10*g* Release 2 (10.1.2.0.2)

The new administrative features of OracleAS Web Cache 10*g* Release 2 (10.1.2.0.2) include:

Enhanced Support for Configuring OracleAS Web Cache Solely as a Software Load Balancer or Reverse Proxy

In previous releases, you could configure OracleAS Web Cache solely as a software load balancer or reverse proxy in place of hardware load balancers.

In this release, support for this mode has been enhanced. You can now configure OracleAS Web Cache as software load balancer or reverse proxy even in front of an application using **Edge Side Includes (ESI)** or in front of another OracleAS Web Cache forming a **cache hierarchy**. A typical OracleAS Portal deployment, for example, has a built-in OracleAS Web Cache used for ESI assembly.

If you require other OracleAS Web Cache features, such as caching or compression support, do not configure this mode. Instead, configure a hardware load balancer or operating system load balancing support, and use OracleAS Web Cache's load balancing feature to manage requests to origin servers.

> **See Also:** "OracleAS Web Cache Solely as a Software Load Balancer or Reverse Proxy" on page 8-42

New Port Numbers for Some Components

To guard against conflicts with Windows ephemeral port number assignments, default port numbers and ranges have been changed for the OracleAS Web Cache operations ports.

> **See Also:** "Using the Default Configuration" on page 8-1

New Features in 10*g* Release 2 (10.1.2.0.0)

The new features for OracleAS Web Cache in 10*g* Release 2 (10.1.2.0.0) include:

Oracle Enterprise Manager 10*g* Application Server Control Support

In previous releases, you had to use OracleAS Web Cache Manager to configure OracleAS Web Cache. In this release, you have two choices:

- You can use Oracle Enterprise Manager 10*g* Application Server Control Console to configure OracleAS Web Cache along with other Oracle Application Server components.

- You can continue to use standalone tool OracleAS Web Cache Manager.

  For standalone OracleAS Web Cache installations or installations in which OracleAS Web Cache was not installed as part of Oracle Application Server, OracleAS Web Cache Manager is the only tool provided.

  > **See Also:**
  >
  > - "Overview of Tools for Managing OracleAS Web Cache within Oracle Application Server" on page 6-1
  > - "Overview of Tools for Standalone Configurations" on page 6-11
  > - *Oracle Application Server Administrator's Guide* for further information about the Application Server Control Console

URL Path Prefix in Site Definitions

When configuring a site definition, you can specify a URL path prefix for those sites that share the same host name. For example, to treat `http://www.company.com/employee` and `http://www.company.com/customer` as two distinct sites, you specify `/employee` and `/customer` as prefixes.

> **See Also:**
>
> - "Site Definitions and Site-to-Server Mappings" on page 1-8
> - "Creating Site Definitions" on page 8-28

Caching Rule Enhancements

- Simplified Configuration for Excluding the Value of Embedded URL Or `POST` Body Parameters

  In previous releases, to ignore the value of embedded URL or `POST` body parameters, you had to configure a session definition for the parameter, and then configure a session caching policy. In this release, you only need to configure the name of the parameter to ignore.

  It is still possible to exclude parameters by configuring a session definition with an accompanying session caching policy. If you have a configuration that specifies the same parameter for both exclusion parameter and a session definition with an accompanying session caching policy, the session method takes precedence. To avoid conflicts, Oracle recommends migrating to the new configuration methods to exclude parameters.

- Enabling and Disabling Caching Rules

You can select to enable or disable caching rules. This feature is intended to provide convenience, so that you do need to create, remove, and then re-create ruled during staging and performance diagnosis.

> **See Also:**
>
> - "Excluding the Value of Embedded URL or POST Body Parameters" on page 2-10 for further information about excluding parameters
>
> - "Configuring Caching Rules and Rule Association" on page 12-7 for further information about enabling and disabling caching rules

Improved Diagnostics

- Tracking `Oracle-ECID` Information in the Response

  In 10*g* (9.0.4), OracleAS Web Cache provided the ability to log the request ID and sequence number from the `Oracle-ECID` request header in the event and access logs. The `Oracle-ECID` request header is used to track requests as they move through the Oracle Application Server architecture. OracleAS Web Cache expands support in 10*g* Release 2 (10.1.2.0.2) to include `Oracle-ECID` information whenever you configure to display diagnostic information in the `Server` response-header field or the HTML response body

- Additional Access Logging Formats for Tracking `Oracle-ECID` Information

  In 10*g* (9.0.4), OracleAS Web Cache provided the End-User Performance Monitoring Format that included the `x-ecid` field for tracking the `Oracle-ECID` information. In 10*g* Release 2 (10.1.2.0.2), `Oracle-ECID` information tracking is being expanded to include two new logging formats, Enhanced CLF (ECLF) and Enhanced Combined Log Format.

- Improved Popular Cache Requests Reporting

  The non-cacheable misses reported in the Popular Requests report have been expanded to include methods other than `GET` or `POST` request methods, HTTP status codes of non-200, requests not matching any rule with a query string, or the `POST` body is too large.

> **See Also:**
>
> - "Server Response-Header Field" on page 2-22 and "Displaying Diagnostic and Event Log Information in the HTML Body or Server Response-Header Field" on page 15-11 for further information about configuring diagnostic output in the `Server` response-header field or the HTTP response message that includes `Oracle-ECID` information
>
> - "Format of the Access Log Files" on page 15-14 for further information about the new access log formats
>
> - "Listing Popular Requests and Cache Contents" on page 15-1 for further information about the Popular Requests report

xx

Part I

Getting Started with OracleAS Web Cache

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

This part contains the following chapters:

- Chapter 1, "Introduction to OracleAS Web Cache"
- Chapter 2, "Caching Concepts"
- Chapter 3, "Cache Clustering"
- Chapter 4, "OracleAS Web Cache Security"
- Chapter 5, "OracleAS Web Cache Topologies"
- Chapter 6, "Introduction to Administration Tools"
- Chapter 7, "Starting and Stopping OracleAS Web Cache"

1

Introduction to OracleAS Web Cache

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

This chapter contains these topics:

- What Is the Big Picture for Caching?
- Oracle's Solution to Web Site Performance Issues
- How Reverse Proxy Web Caching Works
- Benefits of Web Caching
- Features of OracleAS Web Cache
- Compatibility with Oracle Application Server Components

What Is the Big Picture for Caching?

The Web-based computing model creates new performance challenges for application developers and administrators. To carry out e-business successfully, Web sites must protect against poor response time and system outages caused by peak loads. Slow performance translates into lower productivity and lost revenue.

Many administrators try to counter this problem by adding more **application Web server**s to their existing architecture. As more users access these Web sites, more and more servers will have to be added. In short, the hardware and manageability costs associated with adding servers often outweigh the benefits.

Static caches and content distribution services can provide some relief. However, these solutions are unable to serve content that is dynamically generated. Developers spend time writing custom caching solutions.

Oracle's Solution to Web Site Performance Issues

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

By storing frequently accessed URLs in memory, OracleAS Web Cache eliminates the need to repeatedly process requests for those URLs on the application Web server and database tiers. Unlike legacy proxies that handle only static objects, OracleAS Web Cache caches both static and dynamically generated content from one or more

application Web servers. Because OracleAS Web Cache is able to cache more content than legacy proxies, it provides optimal performance by greatly reducing the load on application Web server and database tiers. As an external cache, OracleAS Web Cache is also an order of magnitude faster than object caches that run within the application tier.

Figure 1–1 shows the basic architecture. OracleAS Web Cache sits in front of application Web servers, caching their content, and providing that content to clients that request it. When Web browsers access the Web site, they send **HTTP protocol** or **HTTPS protocol** requests to OracleAS Web Cache. OracleAS Web Cache, in turn, acts as a virtual server on behalf of the application Web servers. If the requested content has changed, OracleAS Web Cache retrieves the new content from the application Web servers. The application Web servers may retrieve their content from an Oracle database. OracleAS Web Cache can be deployed on its own dedicated tier of computers or on the same computer as the application Web servers.

Figure 1–1 OracleAS Web Cache Architecture
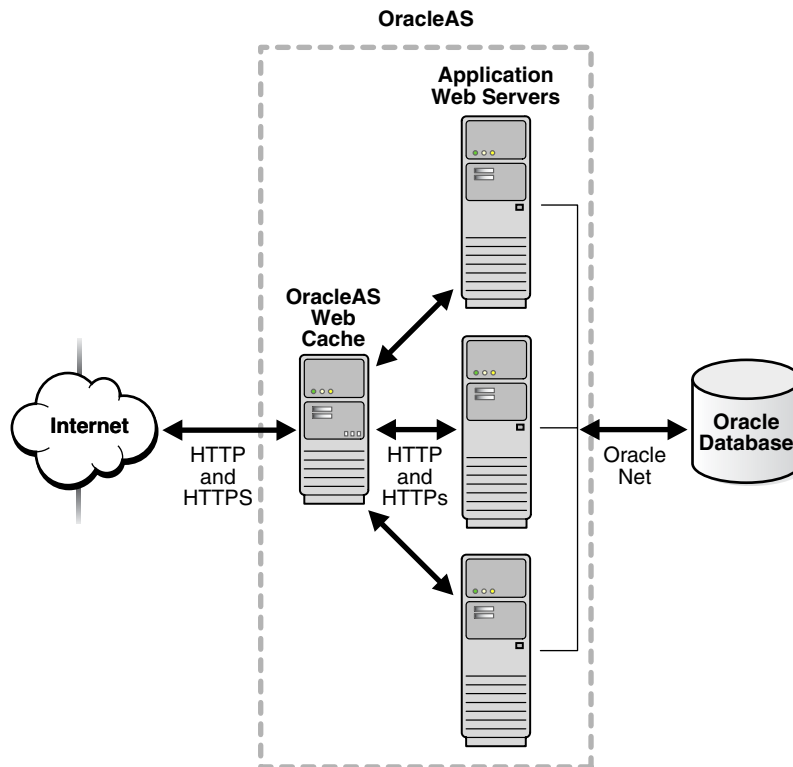
> **Note:** OracleAS Web Cache is compatible with Oracle HTTP Server or any other HTTP-compliant application Web server. See Appendix D for further information about third-party application Web server support.

How Reverse Proxy Web Caching Works

As a reverse proxy, OracleAS Web Cache acts a gateway to the backend servers. Figure 1–2 on page 1-3 shows how OracleAS Web Cache acts a reverse proxy cache for a backend application Web server. OracleAS Web Cache has an IP address of 144.25.190.241 and the application Web server has an IP address of

`144.25.190.242`. The steps for browser interaction with OracleAS Web Cache follow:

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

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

2. DNS returns the IP address of the load balancer for the site, that is, `144.25.190.240`.

3. The browser sends the request for a Web page to the load balancer. In turn, the load balancer sends the request to OracleAS Web Cache server `144.25.190.241`.

4. If the requested content is in its cache, then OracleAS Web Cache sends the content directly to the browser. This is called a **cache hit**.

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

6. The application Web server sends the content to OracleAS Web Cache.

7. OracleAS Web Cache sends the content to the client and stores a copy of the page in cache.

Note: OracleAS Web Cache marks an object stored in the cache for removal when it becomes invalid or outdated, as described in "About Cache Consistency and Performance Assurance" on page 2-1.

Figure 1–2   Web Server Acceleration
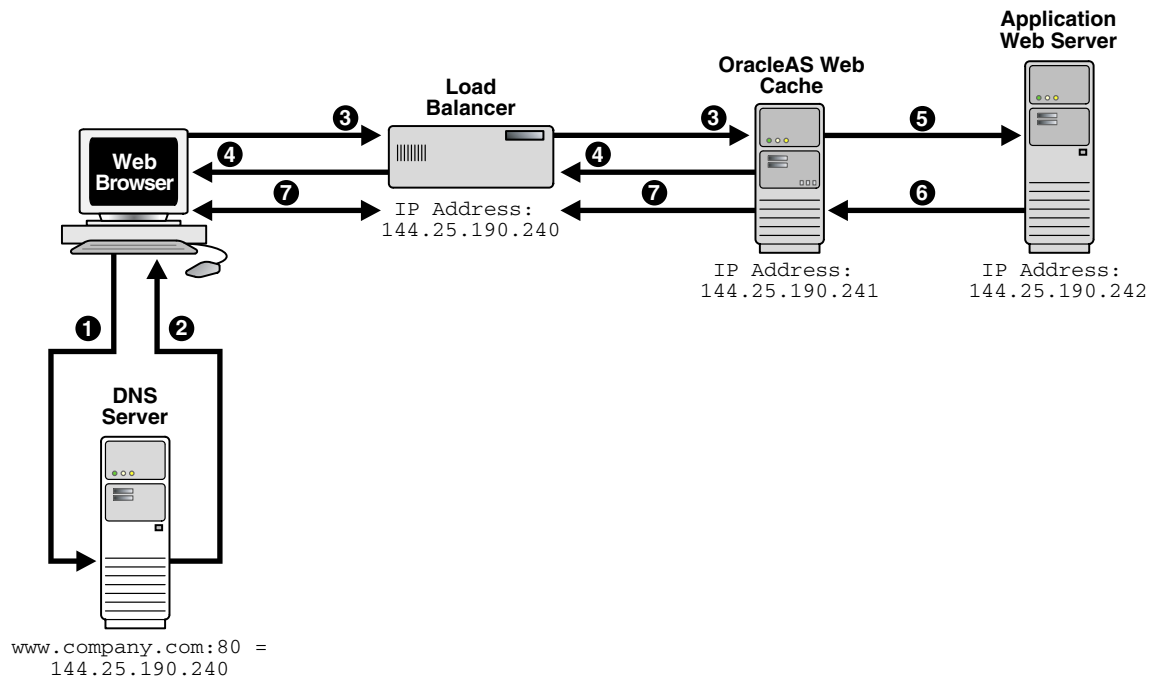
If you do not require caching support, you can configure OracleAS Web Cache solely as a software load balancer or reverse proxy. You configure OracleAS Web Cache settings for this mode.

> **See Also:** "OracleAS Web Cache Solely as a Software Load Balancer or Reverse Proxy" on page 8-42

Benefits of Web Caching

Web caching provides the following benefits for Web-based applications:

- Performance

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

- Scalability

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

- High availability

  OracleAS Web Cache supports load balancing and failover detection for application Web servers. These features ensure that cache misses are directed to the most available, highest-performing Web server in the server farm. Moreover, a patent-pending capacity heuristic guarantees performance and provides surge protection when the application Web server load increases.

- Cost savings

  Better performance, scalability and availability translates into cost savings for Web site operators. Because fewer application Web servers are required to meet the challenges posed by traffic spikes and denial of service attacks, OracleAS Web Cache offers a simple and inexpensive means of reducing a Web site's cost for each request.

- Developer productivity

  Application developers can use OracleAS Web Cache to cache content rather than design and develop application-specific caches.

Features of OracleAS Web Cache

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

- Whole-Page Caching for Static and Dynamic Content Caching
- Cache Invalidation and Expiration
- Performance Assurance
- Virtual Hosting
- Cache Clustering
- Cache Hierarchies
- Origin Server Surge Protection, Load Balancing, Failover, and Session Binding
- Security

- Compression
- Auto-Restart

Whole-Page Caching for Static and Dynamic Content Caching

Caching rules determine which objects OracleAS Web Cache caches. Rules fall into three categories:

- Rules for static content, such as GIF, JPEG, or static HTML files.
- Rules for dynamically generated content created using technologies like JavaServer Pages (JSP), Active Server Pages (ASP), PL/SQL Server Pages (PSP), Java servlets, PHP Hypertext Preprocessor (PHP), and Common Gateway Interface (CGI). Support of these technologies enables OracleAS Web Cache to recognize rules for the following:
  - Multiple-version objects for the same URL, that is, the same URL with slightly different content
  - Session-aware rules for pages containing session information
  - Personalization rules for pages containing personalized greetings, such as "Welcome *Name*," and **session-encoded URLs**
- Pages that require personalized content assembly of dynamic **Edge Side Includes (ESI)** fragments.

> **See Also:**
>
> - "How OracleAS Web Cache Can Cache Dynamically Generated Content" on page 2-3 for further information about dynamically-generated content
> - "About Edge Side Includes (ESI) for Partial Page Caching" on page 2-13

Cache Invalidation and Expiration

OracleAS Web Cache supports **invalidation** as a mechanism to keep the cache consistent with the content on the application Web servers. Administrators can invalidate cache content in one of three ways:

- Manual methods, including OracleAS Web Cache Manager or `telnet`
- Automatic methods with database triggers, scripts, or application logic
- Inline invalidation implemented as part of ESI template pages

When objects are invalidated and a client requests them, OracleAS Web Cache refreshes them with new content from the application Web server.

In addition to invalidation, OracleAS Web Cache provides **expiration**, whereby an expiration time limit is assigned to the objects. When an object expires, OracleAS Web Cache treats it like an invalid object. If the object is requested by a client, OracleAS Web Cache refreshes it with updated content from the application Web server.

> **See Also:** "About Cache Consistency and Performance Assurance" on page 2-1 for further information about invalidation and expiration

Performance Assurance

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

To handle performance issues while maintaining cache consistency, OracleAS Web Cache uses built-in **performance assurance heuristics** that enable it to assign a queue order to objects. These heuristics determine which objects OracleAS Web Cache can serve stale and which objects to refresh immediately. OracleAS Web Cache refreshes objects with a higher priority first and objects with a lower priority at a later time.

OracleAS Web Cache bases the queue order of objects on **popularity** and **validity** assigned during invalidations. If the current load and capacity of the application Web server is not exceeded, then OracleAS Web Cache refreshes the most popular and least valid objects first.

> **See Also:** "About Cache Consistency and Performance Assurance" on page 2-1 for further information about performance assurance

Virtual Hosting

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

This section covers the following topics:

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

Virtual Host Sites

Virtual host sites are sites hosted by OracleAS Web Cache. In other words, clients can request cached content from these sites through OracleAS Web Cache. Figure 1–3 on page 1-7 shows OracleAS Web Cache caching content for two sites, www.1st.company.com and www.2nd.company.com. An additional mapping of www.*.company.com uses *, enabling additional virtual sites that map to host4 and host5 to be added. In addition to caching content, OracleAS Web Cache can also assemble ESI fragments from these sites.

Figure 1–3   Multiple Virtual Host Sites
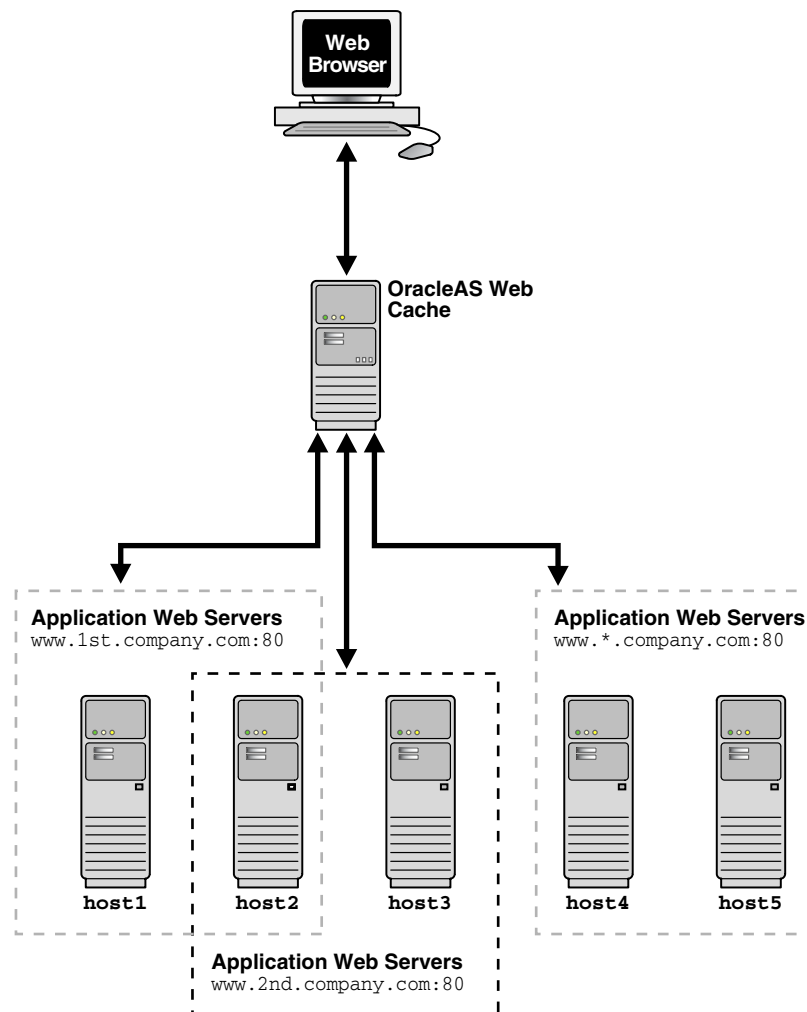
ESI Provider Sites

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

Figure 1–4 on page 1-8 shows an ESI provider site configuration. In this configuration, OracleAS Web Cache receives a request for a page with ESI markup tags. OracleAS Web Cache sends the request to the application Web server. The application Web server uses an application to create a template page and sends it back to OracleAS Web Cache for assembly. OracleAS Web Cache includes the ESI fragments for the template page either with a cached copy of the contents or by contacting the following for that fragment's contents:

- Provider site `www.providersite1.com`

- Another OracleAS Web Cache server, which is caching content for `www.providersite2.com`

> **See Also:** "About Edge Side Includes (ESI) for Partial Page Caching" on page 2-13 for further information about ESI

Figure 1–4   Multiple ESI Provider Sites
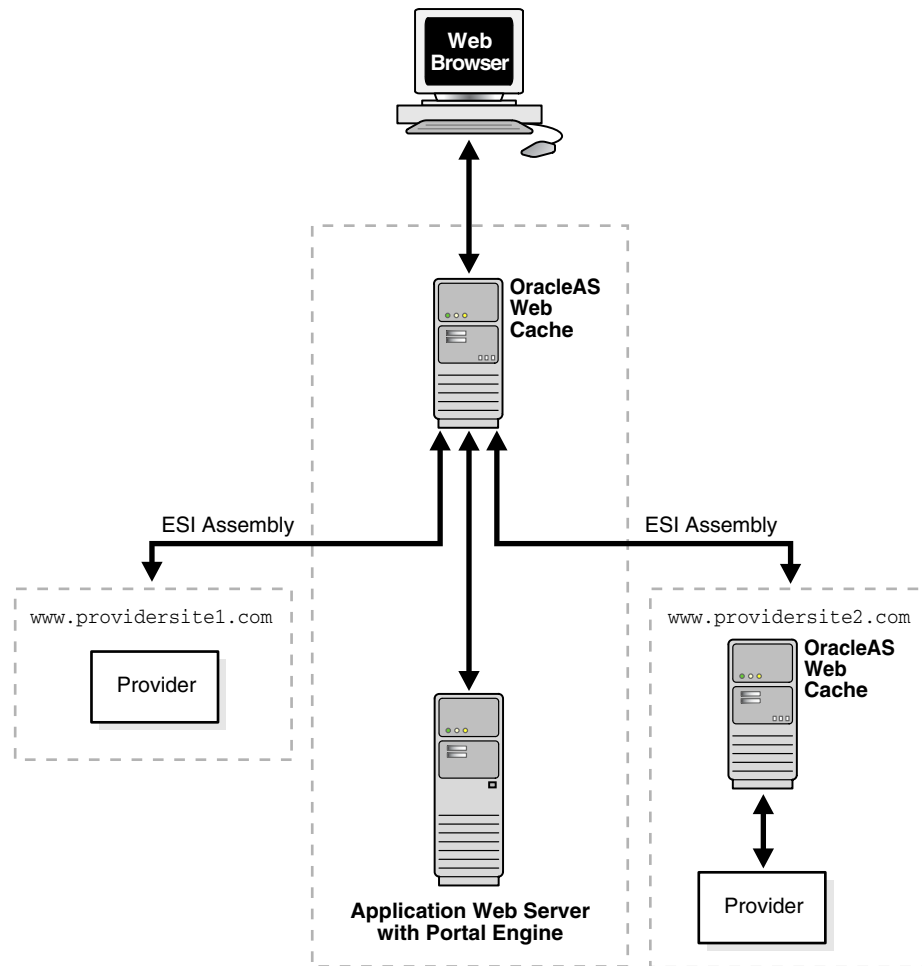
Site Definitions and Site-to-Server Mappings

In order for OracleAS Web Cache to recognize a virtual host site or an ESI provider site, administrators need to perform the following steps:

1. Create a site definition that includes all the names and listening port numbers of the site that clients specify.

   If two sites share the same host name, specify a URL path prefix to distinguish the sites.

2. Determine if OracleAS Web Cache communicates with application Web servers or **proxy server**s.

   OracleAS Web Cache uses application Web servers for internal sites and proxy servers for external sites outside a firewall.

3. Specify the host name and listening port number of the application Web server or proxy server.

4. To ensure requests are sent to the appropriate server, map the site to the application Web servers or proxy servers.

5. Order the mappings. For each request, the first matching mapping is used.

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

- Because requests are resolved to the first matching mapping, give mappings that contain the optional URL path prefix a higher priority than those mappings without an URL path prefix.

  For example, you should order the following mappings as follows:

  ```
  http://www.company.com/portal/page?_pageid=33,4232&_dad=portal
  http://www.company.com/um/traffic_cop?mailid=inbox
  http://www.company.com
  ```

  If you instead reorder the mappings as follows, the request for URLs `http://www.company.com/portal/page?_pageid=33,4232&_dad=portal` and `http://www.company.com/um/traffic_cop?mailid=inbox` will never be reached. Requests for these URLs will instead resolve to `http://www.company.com` because it is listed first:

  ```
  http://www.company.com
  http://www.company.com/portal/page?_pageid=33,4232&_dad=portal
  http://www.company.com/um/traffic_cop?mailid=inbox
  ```

6. Create caching rules that apply to the site and global rules that apply to all sites.

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

To further understand the mappings, reconsider Figure 1–3 on page 1-7. Web sites `www.1st.company.com:80` and `www.2nd.company.com:80` can have site aliases of `1st.company.com:80` and `2nd.company.com:80`, respectively. The site to application Web server mappings are as follows:

- `www.1st.company.com` maps to application Web servers `host1` and `host2`

- `www.2nd.company.com` maps to application Web servers `host2` and `host3`

- `www.*.company.com` maps to `host4` and `host5`

How OracleAS Web Cache Locates Application Web Severs or Proxy Servers

When OracleAS Web Cache receives a client request for an object, it determines the destination site using one of the following elements:

- `Host` request-header field from the request

- Host portion of the requested URL

- `src` attribute of the ESI `<esi:include>` tag

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

If the request does not include host information, then OracleAS Web Cache sends the request to the default site. The default site definition includes the site host name and port information. After an Oracle Application Server installation, a default site definition that includes the host name and listening port of the Oracle HTTP Server is established. If you choose to configure another site as the default site that includes a URL path prefix, the prefix is excluded in matching requests to the site. For example, if you specify a default site of `www.company.com` and a URL path prefix of `/portal`, requests are matched for `www.company.com`, not `www.company.com/portal`.

See Also:

- Chapter 8 for configuration details
- "Default Site Example" on page 8-28 for default site settings

Cache Clustering

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

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

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

See Also:

- Chapter 3 for more information about cache clusters
- Chapter 10 for configuration details

Cache Hierarchies

Many Web-based applications mirror their Web sites in strategic geographical locations. You can deploy multiple instances of OracleAS Web Cache in a **cache hierarchy** so that a local cache stores content from a cache in a central data center for a local market. This enables caches serving local requests to shorten response time and reduce bandwidth and rack space costs for the content provider.

OracleAS Web Cache provides supports two kinds of cache hierarchies:

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

In place of an individual OracleAS Web Cache server in a hierarchy, you can deploy a cache cluster.

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

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

Figure 1–5 on page 1-11 depicts a distributed cache hierarchy. A central cache resides in the United States office and a remote cache resides in Japan. While the central cache in United States stored content retrieved from an application Web server, the remote cache in Japan stores retrieved content from the central cache.

Figure 1–5   Distributed Cache Hierarchy
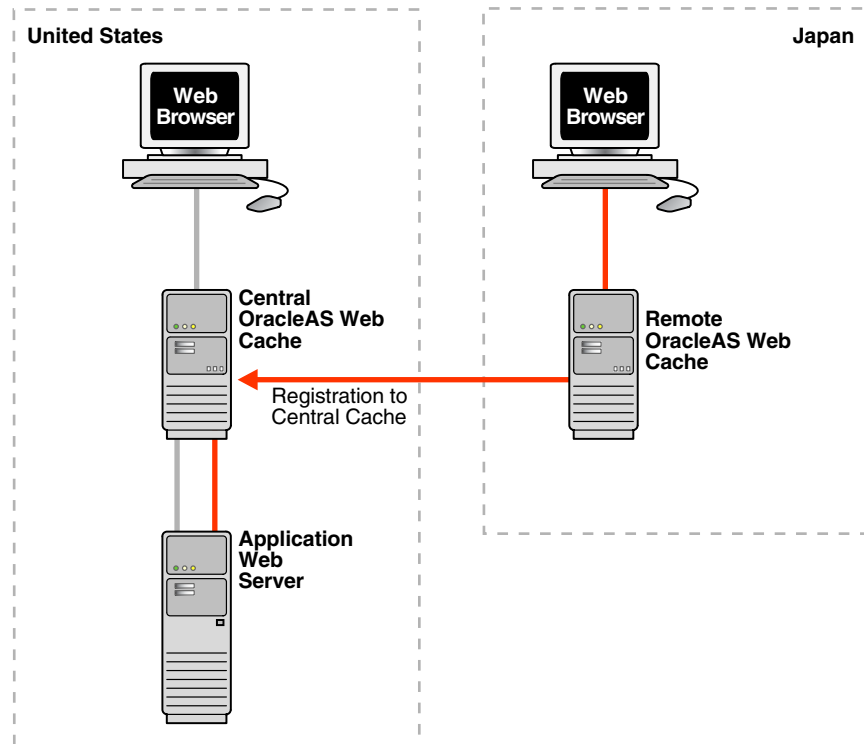
In an ESI cache hierarchy, the subscriber cache is configured with the provider caches as its application Web servers. When the subscriber cache requests content from the provider caches for ESI assembly, the subscriber cache identifies itself as an OracleAS Web Cache server to the provider caches during a transparent registration process. After registration is complete, the provider caches establish a hierarchical relationship with the subscriber cache. Registration also enables invalidation messages to be propagated from the provider caches to the subscriber cache.

Figure 1–6 on page 1-12 depicts an ESI cache hierarchy. A subscriber cache performs ESI assembly. Provider caches locally store ESI fragments for **ESI provider site**s `www.providersite1.com` and `www.providersite2.com`. During ESI page assembly, the subscriber cache contacts the provider caches for the ESI fragments. By storing the ESI fragments locally on the provider caches, fragments are stored both by the provider and subscriber caches. This provides for quick page assembly.

Figure 1–6 ESI Cache Hierarchy
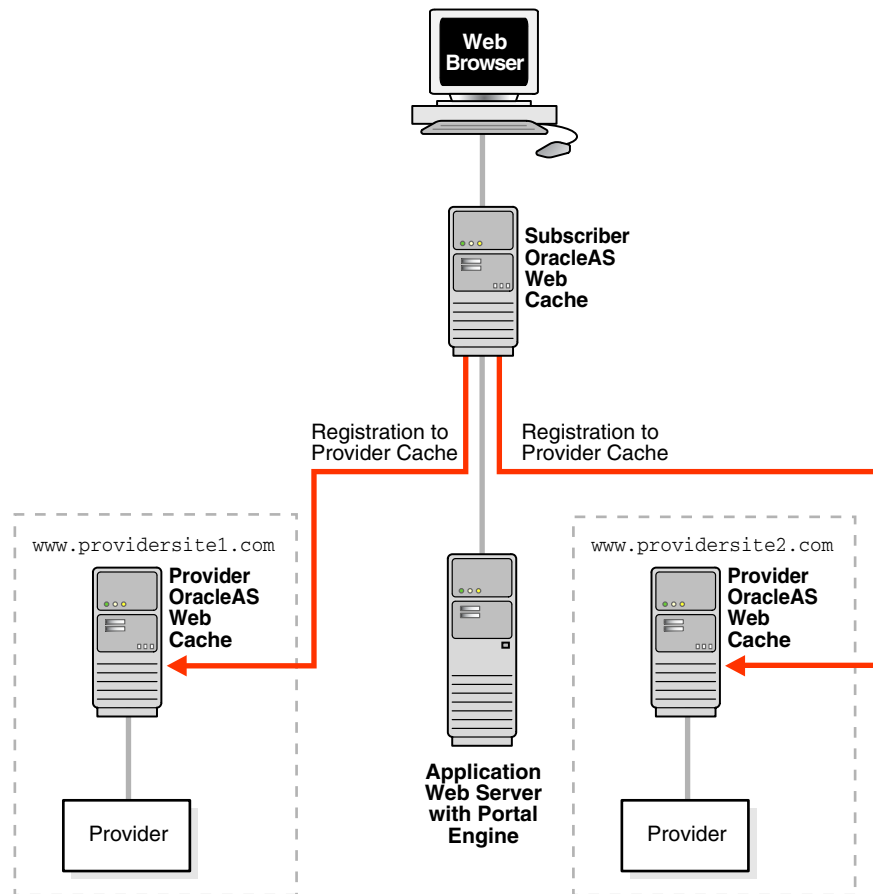
Origin Server Surge Protection, Load Balancing, Failover, and Session Binding

OracleAS Web Cache provides the following features for the application Web server and proxy server it supports:

■ Surge Protection

■ Stateless Load Balancing

■ Backend Failover

■ Session Binding (Stateful Load Balancing)

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

Surge Protection

OracleAS Web Cache passes requests for non-cacheable, stale, or missing objects to origin servers. To prevent an overload of requests on the origin servers, OracleAS Web Cache has a surge protection feature that enables you to set a limit on the number of

concurrent requests that the origin servers can handle. When the limit is reached, subsequent requests are queued. If the queue is full, then OracleAS Web Cache rejects the request and serves a site busy error page to the client that initiated the request.

Stateless Load Balancing

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

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

where:

- `Capacity` is the maximum number of concurrent connections that the origin server can accept

- `Load` is the number of connections currently in use

If the weighted available capacity is equal for multiple origin servers, OracleAS Web Cache sends requests to the origin servers using **round robin**. With round robin, the first origin server in the list of configured servers receives the request, then the second origin server receives the second request. If the weighted available capacity is not equal, OracleAS Web Cache sends the request to the origin server with the most available capacity.

If the load of origin servers is equivalent, OracleAS Web Cache continues to use round robin, even when capacity is not equal for origin servers. Therefore, it is possible to see an even distribution of requests to origin server when the capacities are not configured to be the same.

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

> **See Also:**
>
> - "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25 for instructions on specifying capacity
>
> - "Task 10: Configure Web Site Settings" on page 8-27 for instructions on creating site-to-server mappings

Figure 1–7 on page 1-14 shows two sites, `www.company.com:80` and `www.server.com:80`. The site `www.company.com:80` is supported by application Web servers `company1-host` and `company2-host` with capacities of 50 each. The site `www.server.com:80` is supported by application Web servers `server1-host`, `server2-host`, and `server3-host` with capacities of 150, 50, and 50, respectively.

Figure 1–7   Load Balancing
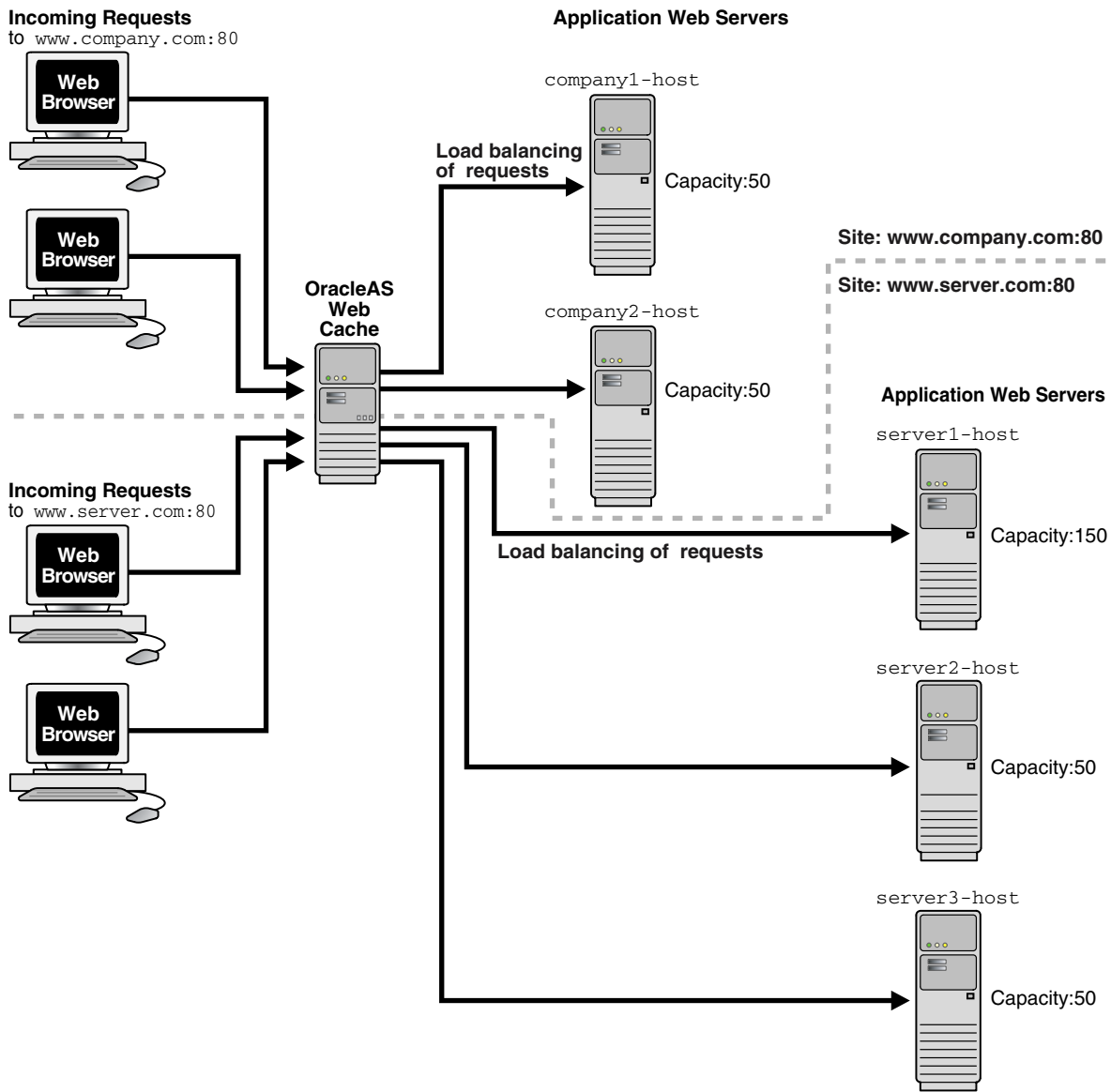
Assuming all application Web servers have an initial load of 0, the requests to `www.company.com:80` and `www.server.com:80` will be distributed in the following manner:

- The requests to `www.company.com:80` are distributed between the two origin servers using round robin.

  The requests to `company1-host` and `company2-host` will be distributed between the two origin servers so that they maintain an equal load. The first request is sent to `company1-host`. The second request is sent to `company2-host` if `company1-host` is still processing the first request. The third and subsequent requests are sent to the origin server that has the highest weighted available capacity.

  When the capacities are equal, OracleAS Web Cache uses round robin to distribute requests.

- The requests to `www.server.com:80` are distributed between three origin servers using the weighted available capacity percentage.

  The first request to `www.server.com:80` is sent to `server1-host`, because it is the first in the configured list. The second request is sent to `server2-host`, because `server1-host` is still processing the first request and has a weighted available capacity of 99.3 percent and `server2-host` has a weighted available capacity of 100 percent. The third request is sent to `server3-host` because `server2-host` is still processing a request and has a weighted available capacity of 98 percent and `server3-host` has a weighted available capacity of 100 percent. The fourth request is sent to `server1-host` because `server2-host` and `server3-host` are still processing requests and have weighted available capacities of 98 percent. The fifth request is sent to `server1-host` because its weighted available capacity is 98.6 percent, which is still greater than `server2-host` and `server3-host`, respectively.

  When the capacities and loads are not equal, OracleAS Web Cache uses the weighted available capacity to distribute requests. If requests were processed before new requests came in, then it is possible for all three origin servers to have loads of 0. In this case, OracleAS Web Cache uses round robin.

If you do not require caching support and need a low-cost solution to a hardware load balancer, you can configure OracleAS Web Cache solely as a software load balancer. This configuration mode is useful for managing traffic to a low-volume, departmental, or test Web site.

> **See Also:**

Backend Failover

After a specified number of continuous request failures, OracleAS Web Cache considers an origin server as failed. When an origin server fails, OracleAS Web Cache automatically distributes the load over the remaining origin servers and polls the failed origin server for its current up or down status until it is back online. Existing requests to the failed origin server result in errors. However, new requests are directed to the other origin servers. When the failed server returns to operation, OracleAS Web Cache includes it in its weighted available capacity to load balance requests.

The **failover** feature is shown in . 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
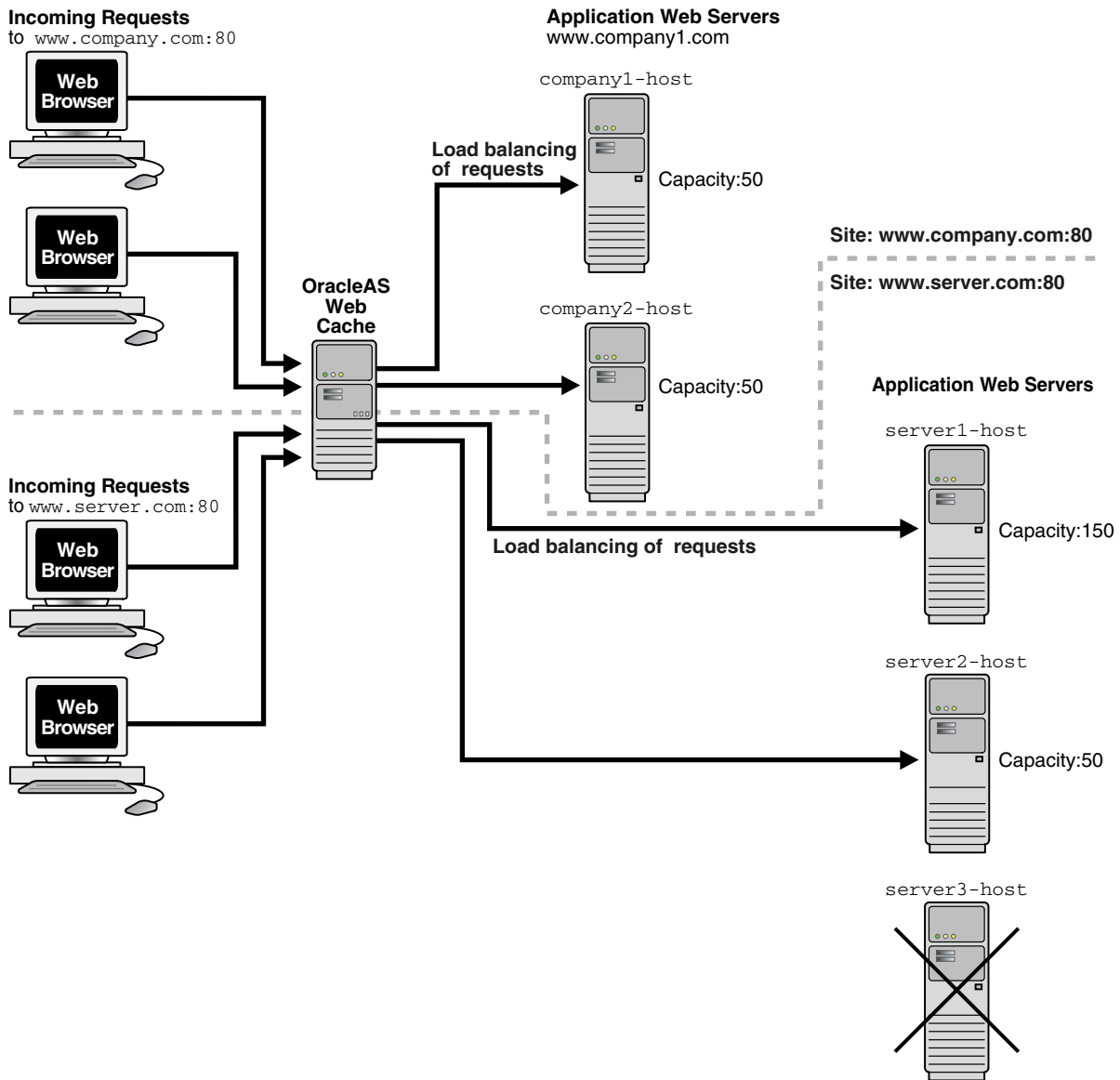
Session Binding (Stateful Load Balancing)

OracleAS Web Cache supports applications that use a session ID or **session cookie** to bind user sessions to a given origin server in order to maintain state for a period of time. To utilize the **session binding** feature, the origin server itself must maintain state, that is, it must be stateful. An application binds user sessions by including session data in the HTTP header or body it sends to a client in such a way that the client is forced to include it with its next request. This data is transferred between the origin server and the client through OracleAS Web Cache either with an **embedded URL parameter** or through a cookie, which is a text string that is sent to and stored on the client. OracleAS Web Cache does not process the value of the parameter or cookie; it simply passes the information back and forth between the origin server and the client.

Figure 1–9 on page 1-17 shows how OracleAS Web Cache supports objects that use session binding.

Figure 1–9   Session Binding
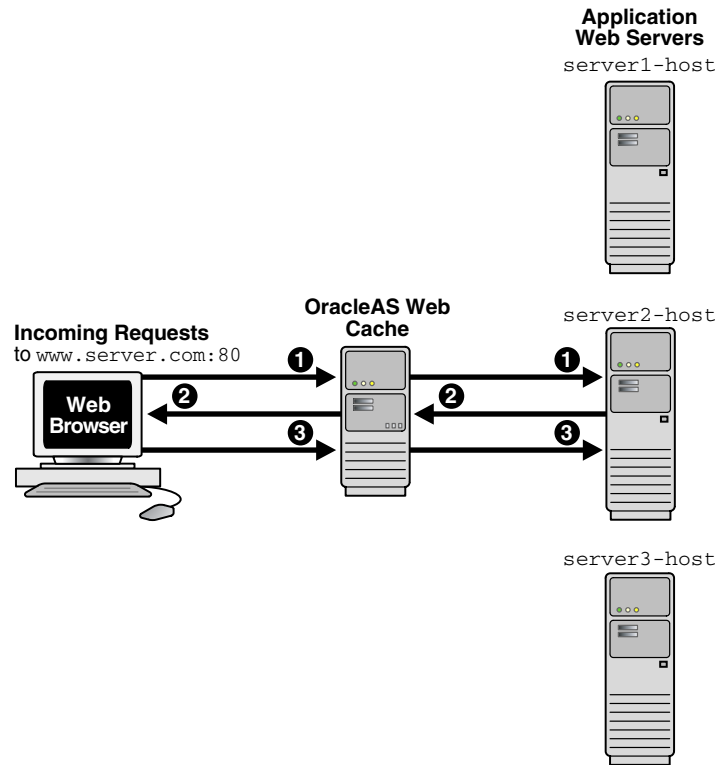
The steps for how session binding works for requests are as follows:

1.   When a request first comes in, OracleAS Web Cache uses load balancing to determine to which origin server the request is forwarded. In this example, application Web server `www.server2.com` is selected.

2.   If the requested object requires session binding, the origin server sends the session information back to the client through OracleAS Web Cache in the form of a cookie or an embedded URL parameter.

3.   OracleAS Web Cache sends subsequent requests for the session to the origin server that established the session, bypassing load balancing. In this example, application Web server `www.server2.com` handles the subsequent requests.

To configure session binding:

1.   Specify a session definition that specifies the name of session cookie or embedded URL parameter.

     You can also choose to use one of the default definitions, described in row Session Definitions of Table 8–1 on page 8-1.

2.   Specify session binding settings.

In a cache cluster, all cache cluster members must be able to determine which origin server established the session, although the request was routed originally through only one cache cluster member. To configure session binding in a cache cluster, you must enable the Session Binding cookie, which is specific to OracleAS Web Cache. OracleAS Web Cache generates the cookie, which tracks session information so that subsequent requests are bound to the origin server that established the session.

> **See Also:** "Bind a Session to an Origin Server" on page 8-39 for
> instructions on configuring session binding

> **Note:** If an origin server cannot accept any more connections because
> of the load, OracleAS Web Cache disables session binding to that
> origin server and attempts to connect to another origin server.

Security

> **See:** Chapter 4, "OracleAS Web Cache Security"

Compression

You can specify that OracleAS Web Cache compress both cacheable and non-cacheable
objects with either OracleAS Web Cache Manager or the `compress` control directive of
the `Surrogate-Control` response-header field.

OracleAS Web Cache correctly handles compression of different types of content and
different types of browsers. For example, OracleAS Web Cache never compresses GIF
or ZIP files even if you enable compression. Similarly, OracleAS Web Cache does not
compress cascading style sheets for the Netscape 4.*x* or Internet Explorer 5.5 browsers.
With these browsers, compressed cascading style sheets can cause background
attributes, such as background images, to not appear in the output.

Because compressed objects are smaller in size, they are delivered faster to browsers
with fewer round-trips, reducing overall **latency**. Compressed content is then
expanded by browsers that support the GZIP compression in the `Accept-Encoding`
request-header field.

In a cache hierarchy, the cache closest to the browser, a remote cache in a distributed
cache hierarchy or a subscriber cache in an ESI cache hierarchy, does not perform any
compression actions on objects. Instead, you configure the cache closest to the origin
server, a central cache in a distributed cache hierarchy or a provider cache in an ESI
cache hierarchy, to compress non-cacheable objects. By configuring compression on
non-cacheable responses from downstream caches, you can ensure content is delivered
more effectively to browsers.

Generally, the remote cache in a distributed cache hierarchy or the subscriber cache in
an ESI cache hierarchy does not forward the `Accept-Encoding` request-header field
to the origin server. However, if a browser request is for a non-cacheable object, the
remote or subscriber cache can send this header to the central or provider cache. In
turn, these caches instead compress the object.

On average, OracleAS Web Cache is able to compress text files by a factor of 4. For
example, 300 KB files are compressed down to 75 KB.

> **See Also:**
>
> - "Configuring Caching Rules and Rule Association" on page 12-7
>   for instructions on configuring compression
>
> - "Using the Surrogate-Control Response Header as an Alternative
>   to Caching Rules" on page 12-30 for instructions on configuring
>   the `Surrogate-Control` response-header field

Auto-Restart

OracleAS Web Cache provides an auto-restart mechanism that checks that the cache is running and automatically restarts the cache if it is not running.

> **See Also:** "Task 3: Configure Auto-Restart Settings" on page 8-10 for instructions on enabling or disabling auto-restart

Compatibility with Oracle Application Server Components

Table 1–1 describes OracleAS Web Cache compatibility with other Oracle Application Server components.

Table 1–1   Compatibility with Other Oracle Application Server Components

Component	Description
Oracle Business Intelligence Discoverer	OracleBI Discoverer is closely integrated with OracleAS Web Cache to improve Discoverer Viewer's overall scalability, performance, and availability. OracleBI Discoverer uses ESI `Surrogate-Control` headers to govern cacheability of other non-configured responses. Because of this integration, the load on mid-tier and database servers in OracleBI Discoverer deployments is reduced, more Discoverer Viewer users are able to access the system concurrently, and those users experience significantly better response times for workbook operations and common business intelligence queries. **See Also:** *Oracle Business Intelligence Discoverer Configuration Guide*
Oracle Application Server Forms Services	You can deploy OracleAS Web Cache as a load balancer with OracleAS Forms Services applications. **See Also:** *Oracle Application Server Forms Services Deployment Guide*
Oracle Application Server Portal	OracleAS Web Cache has been closely integrated with OracleAS Portal to improve its overall scalability, performance, and availability. OracleAS Portal ships with a number of pre-defined caching and invalidation policies that ensure optimal use of OracleAS Web Cache. OracleAS Web Cache controls have been built into the OracleAS Portal administrative user interface and can also be specified by content providers through the Portal Developer Kit (PDK). **See Also:** Sections 5.7, "Configuring OracleAS Web Cache Caching in OracleAS Portal" and 5.8 "Configuring OracleAS Portal to Use a Dedicated OracleAS Web Cache Instance," in *Oracle Application Server Portal Configuration Guide* for configuration details
Oracle Application Server Single Sign-On	Applications that use OracleAS Single Sign-On can use OracleAS Web Cache. You can configure OracleAS Web Cache to cache content for Oracle HTTP Servers running Single Sign-On partner applications. While you should not cache content from OracleAS Single Sign-On servers for security reasons, you can configure OracleAS Web Cache as a software load balancer in front of multiple Single Sign-On mid-tiers. **See Also:** ■ "Leveraging Oracle Identity Management Infrastructure" on page 4-7 ■ *Oracle Application Server Single Sign-On Administrator's Guide*

Table 1–1 (Cont.) Compatibility with Other Oracle Application Server Components

Component	Description
Oracle Application Server Wireless	OracleAS Wireless is integrated with OracleAS Web Cache to improve page rendering performance and scalability. It should be noted that OracleAS Web Cache does not understand WAP and is not used by OracleAS Wireless in the traditional sense in that the cache does not "front-end" the wireless server. Instead, the cache is used as a repository for post-transformed content; the wireless runtime determines what content needs to be inserted into the cache and when to expire content in the cache. OracleAS Web Cache, in this case, acts as a device adaptation cache rather than a reverse-proxy cache. Since markup content is cached using OracleAS Web Cache, the performance and scalability benefits are due to two factors—reduced device adaptation costs and significantly reduced adapter invocation costs. The savings in terms of device adaptation costs stem from the fact that content that can be shared across users and sessions is essentially transformed only once (for each logical device) from its Mobile XML format. Secondly, since the content is not generated every time by an adapter, the total adapter invocation cost is significantly reduced for a site that has a large subset of cacheable pages.
	See Also: *Oracle Application Server Wireless Administrator's Guide*

2

Caching Concepts

This chapter explains how OracleAS Web Cache decides what content to store in the cache, how it maintains the consistency of that content, and how it assembles and caches dynamically generated content.

This chapter contains these topics:

- About Cache Population
- About Cache Consistency and Performance Assurance
- How OracleAS Web Cache Can Cache Dynamically Generated Content
- About Edge Side Includes (ESI) for Partial Page Caching
- About Request and Response-Header Fields Important for Caching
- How OracleAS Web Cache Processes Requests with a Range Request-Header Field

About Cache Population

Caching rules determine which objects are cached. When you establish a caching rule for a particular URL, those objects contained within the URL are not cached until there is a client request for them. When a client first requests an object, OracleAS Web Cache sends the request to the **origin server**. This request is a **cache miss**. Because this URL has an associated caching rule, OracleAS Web Cache caches the object for subsequent requests. When OracleAS Web Cache receives a second request for the same object, OracleAS Web Cache serves the object from its cache to the client. This request is a **cache hit**.

When you stop OracleAS Web Cache, the cache clears all objects. In addition, OracleAS Web Cache clears resets statistics.

> **See Also:** "About Caching Rules" on page 12-1 for a description of how OracleAS Web Cache determines cache population through caching rules

About Cache Consistency and Performance Assurance

Consistency and performance are crucial for the reliability of OracleAS Web Cache. The following features ensure consistency between the cache and origin servers:

- Invalidation
- Expiration
- HTTP Cache Validation

- **Performance Assurance Heuristics**

Invalidation

You use invalidation for frequently changing content. With **invalidation**, OracleAS Web Cache marks objects as invalid. When objects are marked as invalid and a client requests them, they are removed and then refreshed with new content from the origin servers. You can choose to remove and refresh invalid objects immediately, or base the removal and refresh on the current load of the origin servers.

> **See Also:** "Propagation of Invalidation Messages" on page 13-2 for further information about when invalidation messages are propagated from one OracleAS Web Cache server to another

Expiration

With **expiration**, OracleAS Web Cache marks objects as invalid after a certain amount of time in the cache. Expirations are useful if you can accurately predict when content will change on an origin server or database. To prevent objects from remaining in the cache indefinitely, Oracle recommends creating expiration policies for all cached objects.

> **See Also:** "Configuring Caching Rules and Rule Association" on page 12-7 and "Configuring Expiration Policies" on page 12-14 for instructions on creating expiration policies

HTTP Cache Validation

OracleAS Web Cache uses HTTP/1.1 validation models to determine how to best serve a response to clients. Validation works by the comparing two validators, one in the request header and the other in the cached object's response header, to determine if they represent the same or different entities. Specifically, OracleAS Web Cache uses the `If-Modified-Since` and `If-None-Match` for cache hits.

> **Note:** OracleAS Web Cache does not support weak validators for the `If-None-Match` validator. OracleAS Web Cache supports all other `If-None-Match` request-header field formatting.

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
- Chapter 13, "Sending Invalidation Requests" for instructions on invalidating content
- "Configuring Expiration Policies" on page 12-14 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 clients. For this reason, OracleAS Web

Cache intelligently serves some of the objects stale until the origin servers have the capacity to refresh them.

OracleAS Web Cache provides minimal trade-off between performance and consistency through **performance assurance heuristics** that determine which objects 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.

- 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 OracleAS Web Cache with a logical queue of content to update from the origin servers.

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

How OracleAS Web Cache Can Cache Dynamically Generated Content

Most Web pages today are dynamically generated before delivery to the client browser. Web developers frequently use technologies like JavaServer Pages (JSP), Active Server Pages (ASP), PERL, PL/SQL Server Pages (PSP), servlets, PHP Hypertext Preprocessor (PHP), and Common Gateway Interface (CGI) to design their applications. 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 processed successful bid

- Search results, which can change as you add and remove application data

Because of invalidation, OracleAS Web Cache knows which objects are valid and which objects are invalid. Invalidation is especially important for dynamically generated content that changes frequently.

Most traditional 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 products and services to know when

content has changed. OracleAS Web Cache, on the other hand, supports several mechanisms for receiving invalidation messages from the origin server containing the original content.

For dynamically generated pages, clients 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 clients and origin servers to share information, such as session information, in message headers that browsers pass with every request to the origin server. This message header can contain a `Set-Cookie` request-header field that specifies a **cookie** and its value:

```
Set-Cookie:cookie_name=value
```

Cookies are stored on the client's file system and are often used for identifying users who revisit Web sites. Clients 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 because of privacy concerns. For this reason, applications often embed parameter information in the URL or `POST` body. OracleAS Web Cache accepts requests that use the following characters as delimiters for an **embedded URL parameter**, or **POST body parameter**: question mark (?), ampersand (&), dollar sign ($), or semi-colon (;).

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

OracleAS Web Cache recognizes cookies, embedded URL parameters, and `POST` body parameters, enabling you to configure caching rules for objects with the following characteristics:

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

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

Multiple Versions of the Same Object

Some pages have multiple versions, enabling categorization. Figure 2–1 on page 2-5 shows the same object, `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 2–1 Multiple-Version Object
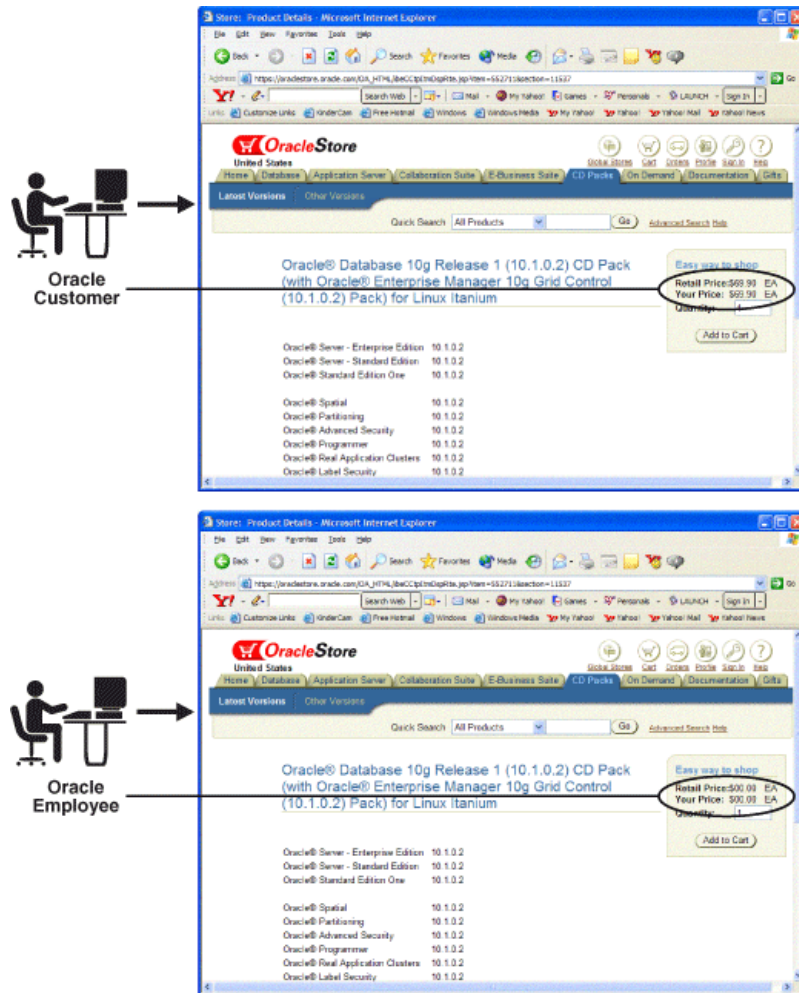
You can configure OracleAS Web Cache to recognize and cache multiple-version pages by using the:

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

For those objects 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 object that do not use the cookie.

When a client sends an initial request for a multiple-version object, OracleAS Web Cache passes the request to the origin server. In its response, the origin server includes a `Set-Cookie` response-header with the category cookie and its value:

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

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

```
Cookie:cookie=value
```

OracleAS 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 its value do not match, then the response is not cached. After versions of the object are cached, OracleAS Web Cache uses the value of the cookie in the client's request to serve the appropriate version of the object to the client browser.

> **Note:** OracleAS 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 client requests that contain cookie values of `Customer`, `Internal`, and `Promotional`. You can configure OracleAS Web Cache to recognize the `user_type` cookie, enabling OracleAS Web Cache to cache three different objects. In addition, you can configure OracleAS Web Cache to cache a fourth object for those requests that do not use a cookie.

Table 2–1   Multiple-Version Object 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 objects that have different versions based on HTTP request headers, configure caching rules that specify the HTTP request header. HTTP request headers enable clients to pass additional information about the request and about themselves. OracleAS Web Cache uses the header to serve the appropriate version of the URL to clients.

OracleAS Web Cache supports all valid HTTP request headers. Table 2–2 lists the HTTP request-header fields supported by the Application Server Control Console and OracleAS Web Cache Manager interfaces, as described in "Configuring Caching Rules and Rule Association" on page 12-7. You can specify other HTTP request-header fields with the `Surrogate-Control` response-header field, as described in "Configuring Expiration Policies" on page 12-14.

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

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

Header Field	Description
`User-Agent`	Contains information about the client that initiated the request
	Example: `User-Agent: Mozilla/4.61 [en] (WinNT; U)`

Note: By default, OracleAS Web Cache does not interpret the values of these HTTP request headers. If the values for two pages are different, OracleAS Web Cache caches both pages separately.

This issue is especially problematic with the `User-Agent` request header, whereby the browser type, version, and operating system can result in multiple cache entries. For example, if one request sends an HTTP request-header field of `User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows)` and another request sends an HTTP request-header field of `User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows; DigExt)` for different versions of Internet Explorer, OracleAS Web Cache serves two pages for the two requests.

You can override this behavior for the `User-Agent` request header by configuring OracleAS Web Cache to cache and serve the same page for the same browser type, as described in "Recognizing Similar Browser Types for Multiple-Version Objects Containing HTTP Request Headers" on page 12-16.

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 OracleAS Web Cache HTML tags `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->`.

OracleAS Web Cache processes these tags and caches the instructions for substituting values for personalized attributes based on the information contained within a cookie, embedded URL parameter, or `POST` body parameter.

This functionality enables OracleAS 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 parameter. All subsequent requests for the page that pass the cookie or parameter are served from the cache.

Note: To achieve personalization within an HTML tag, use ESI, as described in "Using ESI for Simple Personalization" on page 12-34.

Figure 2–2 on page 2-8 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 to the user.

Figure 2–2   Page with a Personalized Attribute
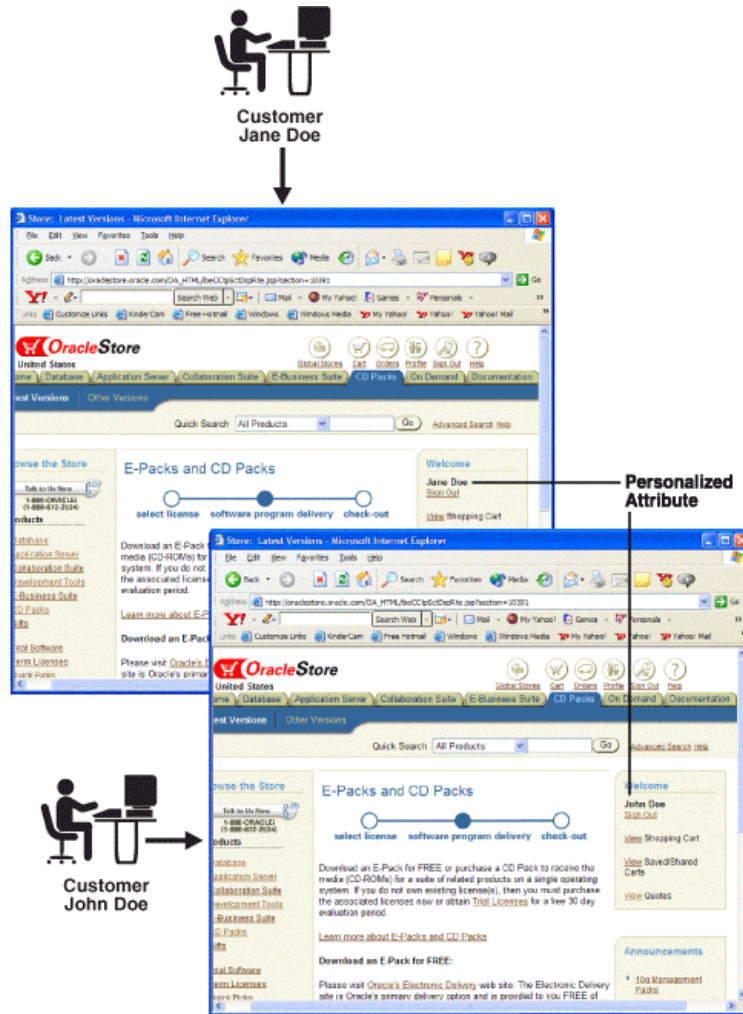
The personalized greeting **Jane Doe** uses the following HTML code:

```
<B>
<!-- WEBCACHETAG="person01"-->
Jane Doe
<!-- WEBCACHEEND-->
</B>
```

The 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 OracleAS 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 OracleAS 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 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 OracleAS Web Cache with a personalized attribute definition of `person_name01` to map to the `person_name` embedded URL parameter. OracleAS Web Cache would then use the value of the embedded parameter to substitute the appropriate name.

If the page supported `POST` body parameters, then the `POST` body for `https://oraclestore.oracle.com/OA_HTML/ibeCZzpHome.jsp` would contain the following code for Jane Doe and John Doe:

```
person_name=Jane+Doe
&cart_items=0
&a=b

person_name=John+Doe
&cart_items=0
&a=b
```

To substitute personalized attribute values:

1. Configure a personalized attribute definition with the personalized attribute cookie, embedded URL parameter, or `POST` body parameter.

2. Configure a caching rule with the option **Session-Encoded URL** enabled. Only requests matching the caching rule will perform the substitution.

If a request does not contain the value of the cookie or parameter, OracleAS Web Cache substitutes the personalized attribute with a default string.

> **See Also:** "Configuring Rules for Popular Pages with Session Establishment" on page 12-29

> **Note:** You can also substitute session values between the
> `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags:
>
> 1. Configure a session definition with the session cookie, embedded URL parameter, or `POST` body parameter.
>
> 2. Configure a caching rule with the option **Session-Encoded URL** enabled. Only requests matching the caching rule will perform the substitution.
>
> **See Also:** "Configuring Rules for Popular Pages with Session Establishment" on page 12-29

Controlling How the Cache Serves Personalized Attribute Requests

You can specify how OracleAS Web Cache serves requests with the existence or nonexistence of personalized-attribute cookies, embedded URL parameters, or `POST` body parameters. You can choose to:

■ Serve or not serve cached objects to requests that have the cookie, embedded URL parameter, or `POST` body parameter

■ Serve or not serve cached objects to requests that do not have the cookie, embedded URL parameter, or `POST` body parameter

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

When you choose to serve for both, you can then specify if requests with or without the cookie or parameter can share the same cached object. OracleAS Web Cache uses a default string for those requests without the cookie or parameter.

To specify how personalized attribute pages are served by OracleAS Web Cache:

1. Configure a personalized attribute definition that specifies the name of the cookie, embedded URL parameter, or `POST` body parameter.

2. Specify the behavior for caching objects 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 or Personalized Attribute Caching Policies" on page 12-19

Session Information

Some Web sites keep track of user sessions by assigning each user a unique session ID. When a client 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.

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

Using session information in a cookie, embedded URL parameter, or `POST` body parameter, you can configure OracleAS Web Cache for the following purposes:

- Excluding the Value of Embedded URL or POST Body Parameters

- Substituting Session Information in Session-Encoded URLs

- Controlling How Session Requests Are Served by the Cache

Excluding the Value of Embedded URL or POST Body Parameters

By default, OracleAS Web Cache distinguishes origin server responses by the request URLs. However, if the request contains an embedded URL or `POST` body parameter, the request URL to the same page content is distinct for each session. Therefore, OracleAS Web Cache caches responses for each of the distinct URLs. This can result in low cache hit rates and redundantly cached objects.

By configuring OracleAS Web Cache to ignore the value of embedded URL or `POST` body parameters, you enable OracleAS Web Cache to serve one cached object to multiple sessions requesting the same page. OracleAS Web Cache caches the response to the first request and serves subsequent requests for the page from its cache.

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
```

User John Doe accesses the same page with a request URL of:

```
https://oraclestore.oracle.com/OA_HTML/ibeCCtpSctDspRte.jsp?section=10103&session_ID=33437
```

In addition, this page contains the following POST body for Jane Doe and John Doe, respectively:

```
section=1013
&session_ID=3346

section=1013
&session_ID=3347
```

The only distinct part to the request URL and the POST body is the value of the session_ID parameter. Rather than caching and serving two versions of the same object, you can configure OracleAS Web Cache to ignore the value of session_ID so that one cached object can be served to both users.

To configure parameters to ignore:

■ Establish global parameters.

■ Specify site-specific parameters overrides.

■ When creating a caching rule, specify either global or site-specific parameters.

> **See Also:** "Configuring OracleAS Web Cache to Exclude the Value of Embedded URL or POST Body Parameters" on page 12-18

Substituting Session Information in Session-Encoded URLs

The section "Excluding the Value of Embedded URL or POST Body Parameters" on page 2-10 describes how you can ignore the value of embedded URL or POST body parameters for objects with identical content for all sessions. However, in some cases, the HTML content of objects 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 OracleAS Web Cache to substitute sessions within HTML hyperlink tags with the session values obtained from a session cookie, embedded URL parameter, or POST body parameter. By configuring session value substitution in combination with ignoring the value of embedded URL parameters, you can configure OracleAS Web Cache to cache one object for multiple sessions, even if the session parameter values in session-encoded URLs vary.

> **Note:** OracleAS Web Cache does not cache the Set-Cookie response header field.

Continuing with the example from "Excluding the Value of Embedded URL or POST Body 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 2–3 on page 2-12 has several <A HREF=...> links that include the session_ID parameter. The Master Index link under the **Oracle Database Standard Edition** heading for Jane Doe uses the following HTML code:

```
<A HREF="http://oraclestore.oracle.com/OA_HTML/
ibeCCtpSctDspRte.jsp?section=10166&session_ID=334326">Master Index</A>
```

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

```
<A HREF="http://oraclestore.oracle.com/OA_HTML/
ibeCCtpSctDspRte.jsp?section=10166&session_ID=334327">Master Index</A>
```

By using the value of the session_ID embedded URL parameter, OracleAS Web Cache substitutes the correct session information for Jane Doe and John Doe.

Figure 2–3   Session-Encoded URLs
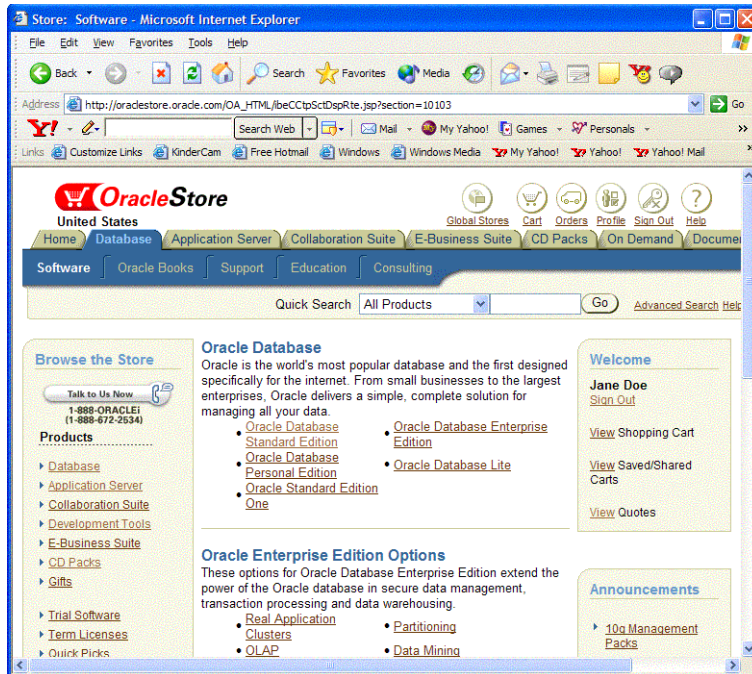
After the cache is populated with a page that contains session-encoded URLs, other requests for the page are served from the cache, regardless of whether the request has a session cookie, embedded URL parameter, or POST body parameter. If the request does not contain a session cookie or embedded URL parameter, OracleAS 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, embedded URL parameter, or POST body 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 **Session-Encoded URL** enabled. Only requests matching the caching rule will perform the substitution.

> **See Also:** "Configuring Support for Session-Encoded URLs" on page 12-21

Controlling How Session Requests Are Served by the Cache

You can specify how OracleAS Web Cache serves requests with the existence or nonexistence of session cookies, embedded URL parameters, or POST body parameters. You can choose to:

■ Serve or not serve cached objects to requests that have a session cookie, embedded URL parameter, or POST body parameter

- Serve or not serve cached objects to requests that do not have a session cookie, embedded URL parameter, or `POST` body parameter

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

When you choose to serve for both, you can then specify if requests with or without the session cookie or parameter can share the same cached object. OracleAS Web Cache uses a default string for those requests without the cookie or parameter.

To specify how session-related pages are served by OracleAS Web Cache:

1. Configure a session definition that specifies the name of the session cookie, embedded URL parameter, or `POST` body parameter.

2. Specify the behavior for caching objects with or without session information by defining a session-caching policy.

3. Associate URLs with the session-caching policy.

> **See Also:**
> - "Configuring Session or Personalized Attribute Caching Policies" on page 12-19 for configuration details
> - "Configuring Rules for Popular Pages with Session Establishment" on page 12-29 for information about caching popular pages that require session establishment

About Edge Side Includes (ESI) for Partial Page Caching

OracleAS 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 divided into fragments of differing caching profiles. These fragments are maintained as separate elements in the cache. The fragments are assembled into HTML pages as appropriate when requested by end users.

By enabling dynamic assembly of Web pages on OracleAS 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 OracleAS Web Cache when requested. Furthermore, page assembly can be conditional, based on information provided in HTTP request headers or end-user cookies.

The section contains the following topics:

- Page Assembly Components
- Fragmentation with the Inline and Include Tags
- Referer Request-Header Field
- Cookie Management for Template Pages and Fragments
- ESI Features
- ESI for Java (JESI)
- Oracle JDeveloper Support for ESI and JESI

Page Assembly Components

The basic structure that an application developer uses to create content for partial-page caching is a template page containing fragments. As depicted in Figure 2–4, 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 2–4   Template Page
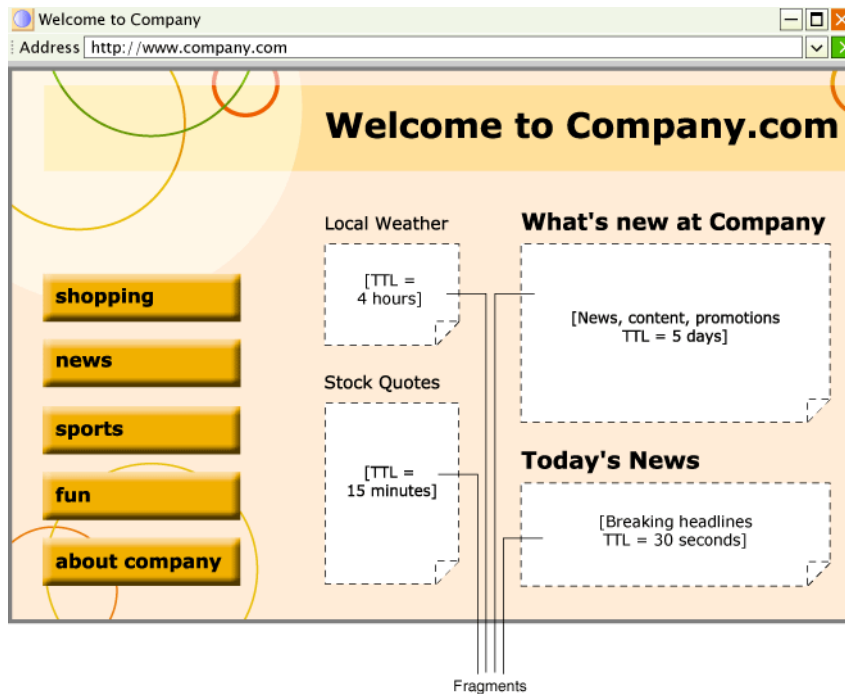
The template page is associated with the URL that end users request. To include the fragments, the template page is configured with ESI markup tags that instruct OracleAS Web Cache to fetch and include the HTML fragments. The fragments themselves are HTML files containing discrete text or other objects.

Each included fragment is a separate object with its own caching policy. 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:comment>`	Specifies comments not be included in the output
`<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

Table 2–3   (Cont.) Summary of ESI Tags

Tag	Description
`<esi:invalidate>`	Specifies an invalidation request within the response of a browser page
`<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

Example 2–1 shows the ESI markup language for the template page shown in Figure 2–4 on page 2-14.

Example 2–1   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="/CompanyStock.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 version
 of the page.
</esi:remove>
...
</BODY>
```

```
</HTML>
```

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

Example 2–2   GetProfile.jsp XML Response

```
<?xml version=1.0?>
<esi:environment esiversion="ORAESI/9.0.4">
  <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
- Selecting the Fragmentation Mechanism for Your Application

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. OracleAS Web Cache stores the enclosed portions as separate fragments and the original page as a page template without the enclosed fragments. Fragments are shared among templates if their names are identical and they are from the same site.

Example 2–3 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.

Example 2–3   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.

To invalidate non-fetchable fragments, you must invalidate both the template object and the non-fetchable fragments to ensure the fragments are invalidated.

> **See Also:** Chapter 13 for details about sending invalidation requests

Using Inline for Fetchable Fragmentation

`<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`.

Example 2–4 shows an `<esi:inline>` example with a fetchable fragment named `/news101`. A request for the page returns the template page and the fetchable fragment.

Example 2–4   Inline Fetchable Example

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

> **See Also:** "ESI inline Tag" on page 16-23 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 the `<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 make 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, OracleAS 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.*

Example 2–1 on page 2-15 shows ESI markup with `<esi:include>` tags.

Selecting the Fragmentation Mechanism for Your Application

Although both `<esi:include>` and `<esi:inline>` enable OracleAS Web Cache to fetch fragments for the client browser, `<esi:include>` is more robust for performing this task and provides an easy way in which to manage fragments. Because `<esi:include>` affects the application flow, it is best to incorporate `<esi:include>` early in the design phase of an application. For an existing application, `<esi:inline>` is better mechanism because it requires minimal change to your application.

Referer Request-Header Field

When OracleAS Web Cache receives a client request for a template page with a `Referer` request-header field, it forwards the request with the `Referer` request header to the origin server. In turn, the origin server returns fragments to OracleAS Web Cache with the URL of the template as the value for the `Referer` header. This functionality associates the fragment request with the template request.

Cookie Management for Template Pages and Fragments

Session cookie establishment for ESI templates and fragments works much the same way as typical OracleAS Web Cache objects with the following additional features:

- `Cookie` request-header field inheritance

  When a client requests an ESI template page that includes fragments, requests for fragment pages are generated in OracleAS Web Cache. A fragment request inherits 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, OracleAS 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, OracleAS 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 not match the `Host` request-header field in the template, OracleAS 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 2-10 for an overview of `Cookie` and `Set-Cookie` behavior
  >
  > - "Variable Expressions" on page 16-5 for a description of how you can use the `HTTP_COOKIE` variable in ESI markup

ESI Features

ESI can be used with HTML, XML, JSP, ASP, 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. OracleAS Web Cache receives a request for a template page.

  2. OracleAS 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.

  OracleAS Web Cache does not perform character set conversion for non-ESI pages.

- XML conversion to HTML

  OracleAS 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 OracleAS 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.

> **See Also:**
>
> - "Configuring Rules for Content Assembly and Partial Page Caching" on page 12-33
>
> - Chapter 16 and http://www.esi.org for an overview of the ESI language
>
> - *Oracle Application Server Containers for J2EE JSP Tag Libraries and Utilities Reference* for a description of JESI

Oracle JDeveloper Support for ESI and JESI

Oracle JDeveloper provides the ESI Servlet Filter extension, which enables developers to create JSPs with ESI or JESI tags, and test them within the development environment. ESI and JESI are caching standards for developing high-performance J2EE applications. Without the extension, JSPs developed with ESI or JESI will not be rendered properly when previewed in Oracle JDeveloper.

> **See Also:** http://www.oracle.com/technology/products/ias/web_cache/index.html for further information about the ESI Servlet Filter

About Request and Response-Header Fields Important for Caching

OracleAS Web Cache uses the following Oracle-specific HTTP request and response-header fields:

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

Surrogate-Capability Request-Header Field

For each requested object from the cache, OracleAS Web Cache appends a `Surrogate-Capability` request-header field to an object's HTTP request message. The `Surrogate-Capability` request-header serves the following purposes:

- Enables applications to detect OracleAS Web Cache
- Identifies the types of ESI operations that OracleAS Web Cache can perform

> **See Also:** "About the Surrogate-Capability Request Header for Cached Objects" on page 16-3

Server Response-Header Field

For objects sent to clients, OracleAS Web Cache adds diagnostic information to the `Server` response-header field of the HTTP response message.

Using the `Server` response header information, you can determine whether a request was served from the cache or the origin server. For diagnostics purposes, it can be useful to also display this information as a textual string in the HTML response body of an object and in the access logs.

> **See Also:**
>
> - "Displaying Diagnostic and Event Log Information in the HTML Body or Server Response-Header Field" on page 15-11 for instructions on disabling the diagnostics information or displaying the information in the HTML response
>
> - "cs(header_name) and sc(header_name) Access Log Fields" on page 15-21 and "Configuring Access Logs" on page 15-22 for further information about creating a user-defined access log format that includes the `sc(Server)` field

Surrogate-Control Response-Header Field

The `Surrogate-Control` response-header field enables application developers to specify caching attributes of an object. This response-header field enables the application Web server to override the caching rules configured through administrative interfaces Oracle Enterprise Manager 10*g* Application Server Control Console or OracleAS Web Cache Manager.

> **See Also:**
>
> - "About Cache Population" on page 2-1 for a description of how OracleAS Web Cache determines when to use caching attributes from the `Surrogate-Control` and the administrative interfaces
>
> - "Using the Surrogate-Control Response Header as an Alternative to Caching Rules" on page 12-30 for information on how to configure the response header
>
> - "About the Surrogate-Control Response Header for Supporting ESI Language Elements" on page 16-2 for information on configuring ESI

Surrogate-Key Response-Header Field

The `Surrogate-Key` response-header field enables application developers to identify search key strings for a given response object. Search keys are strings that may not appear in the URL, cookies, or HTTP request headers of objects. The intent of the search keys is to provide another criteria for invalidation. In addition to the URL of objects, OracleAS Web Cache administrators can base invalidation on one or more search keys used in the `Surrogate-Key` response-header field of objects in the cache.

> **See Also:** "Using Search Keys in Surrogate-Key Response Header and Invalidation Requests" on page 13-39

How OracleAS Web Cache Processes Requests with a Range Request-Header Field

When the first client request for a multi-part object with an HTTP `Range` request-header field comes in, OracleAS Web Cache sends the request to the origin server. OracleAS Web Cache serves the entire object received from the origin server response to the client, and OracleAS Web Cache caches the entire object for the request. For a subsequent request for the object, OracleAS Web Cache serves only the part requested from the client.

3

Cache Clustering

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

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

This chapter contains the following topics:

- Overview of Cache Clusters
- Benefits of Cache Clusters
- How Cache Clusters Work
- How Cache Content Is Distributed
- Failure Detection and Failover

Overview of Cache Clusters

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

Figure 3–1 on page 3-2 shows an OracleAS Web Cache cluster that contains three cache cluster members. As the figure shows, the cluster members communicate with one another as well as with the application Web servers and with the clients.

Figure 3–1   OracleAS Web Cache Cluster Architecture
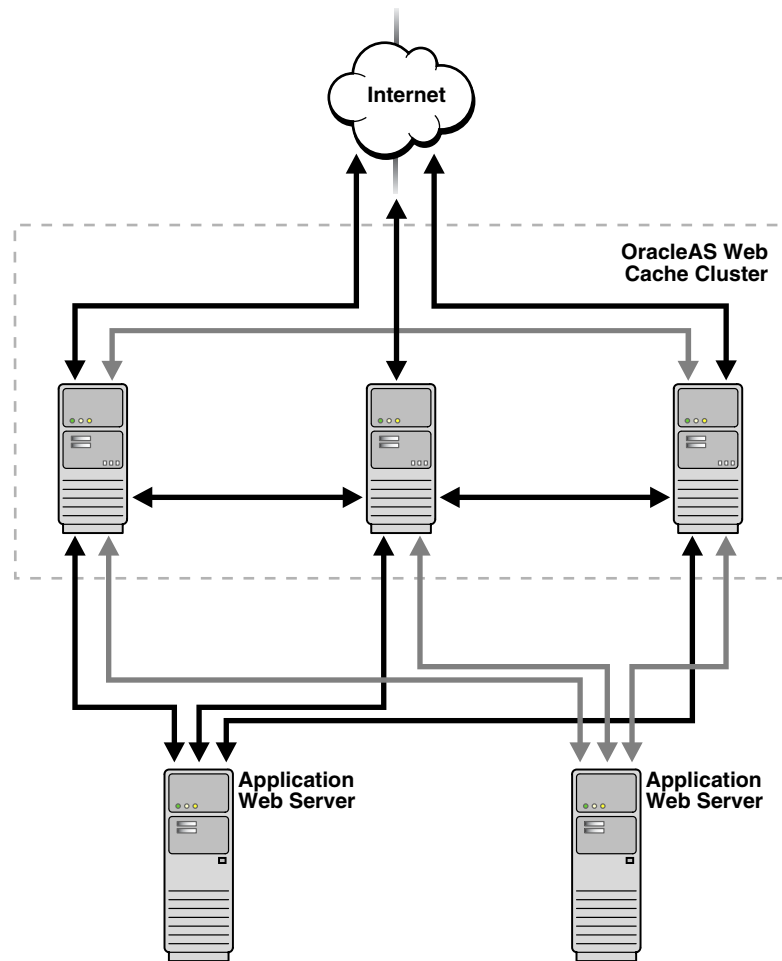
OracleAS Web Cache uses the relative capacity of each cache instance to distribute the cached content among the cache cluster members. In effect, it assigns a cache cluster member to be the owner of a particular object. This content is called **owned content**.

In addition to the owned content, OracleAS Web Cache stores popular objects in the cache of each cluster member. These objects are known as **on-demand content**. By storing the on-demand content, OracleAS Web Cache responds to requests for those objects quickly and decreases the number of cache misses. Fewer requests are sent to the application Web server. The result is improved performance.

Benefits of Cache Clusters

Cache clusters provide the following benefits:

- High availability

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

- Scalability and performance

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

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

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

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

- Reduced load on the application Web server

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

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

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

- Improved data consistency

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

    You can configure a cache cluster that does not support requests between cache cluster members, but allows propagating invalidation requests, as well as propagating configuration changes. See "Configuring Administration and Invalidation-Only Clusters" on page 10-10 for more information.

- Manageability

    Cache clusters are easy to manage because they use one configuration for all cache cluster members. For example, you specify one set of caching rules and one set of invalidation rules. OracleAS Web Cache distributes those rules throughout the cluster by propagating the configuration to each cluster member.

How Cache Clusters Work

In a cache cluster, multiple instances of OracleAS Web Cache operate as one logical cache.

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

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

When you add a cache to the cluster, the cache-specific information of the new cluster member is added to the configuration of the cache cluster. Then, OracleAS Web Cache propagates the configuration to all members of the cluster. Because adding a new member changes the relative capacity of each Web cache, OracleAS Web Cache uses the information about capacity to recalculate which cluster member owns which content.

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

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

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

> **See Also:** "Invalidation in Cache Clusters" on page 13-5 for more information about invalidation propagation in cache clusters

How Cache Content Is Distributed

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

OracleAS Web Cache maintains a structure to record ownership of objects. When a request for a cacheable object is received, OracleAS Web Cache uses the structure to assign a cache cluster member to be the owner of the object.

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

When an incoming request for a non-cacheable object is received by one of the cache cluster members, the request is forwarded to the application Web server.

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

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

- If the content is cached by cache_Y, cache_Y returns the content to the client, as shown in the following figure. The content could be either owned content or on-demand content.
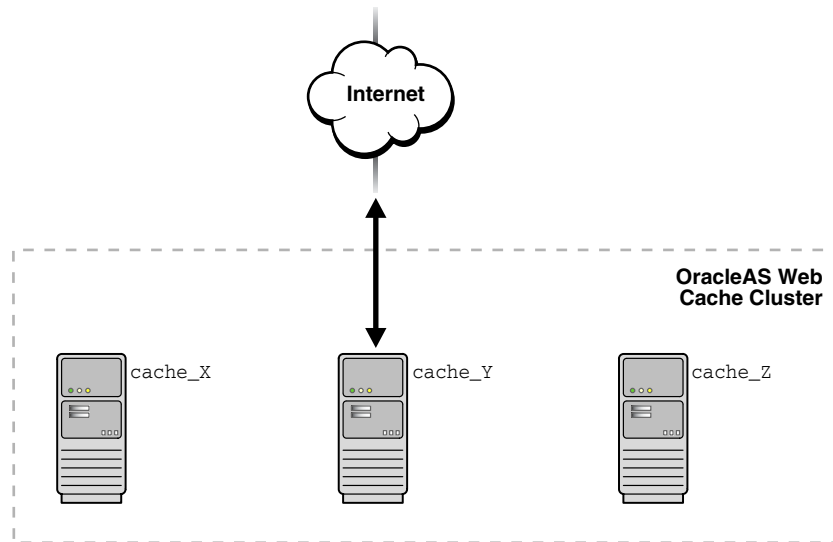
- If the content is not cached by cache_Y, OracleAS Web Cache performs an ownership lookup. Then:

  - If cache_Y is the owner of the requested content, it sends the request to the application Web server, which returns the requested content to cache_Y. Then, cache_Y caches the content and returns it to the client, as shown in the following figure:
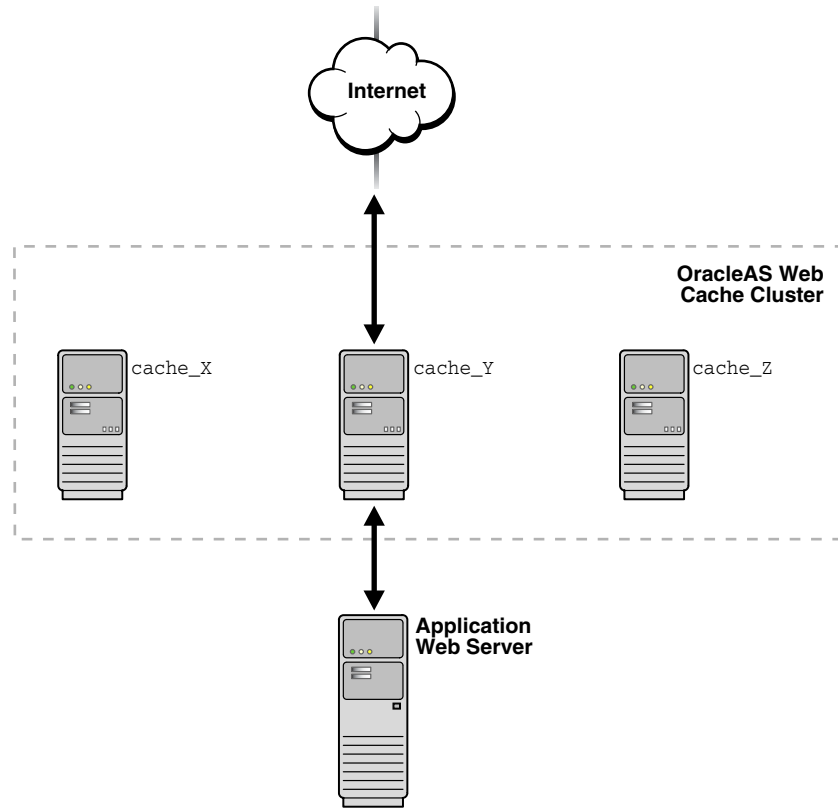
– If cache_Y is the not owner of the requested content, cache_Y sends the request to the owner cache, cache_Z. Then:

* If the requested content is stored in the owner's cache, the owner returns the requested content to the cluster member (cache_Y) that originally received the request. Cache_Y returns the requested content to the client, as shown in the following figure. The content is stored in cache_Y as on-demand content to satisfy future requests for that object quickly.
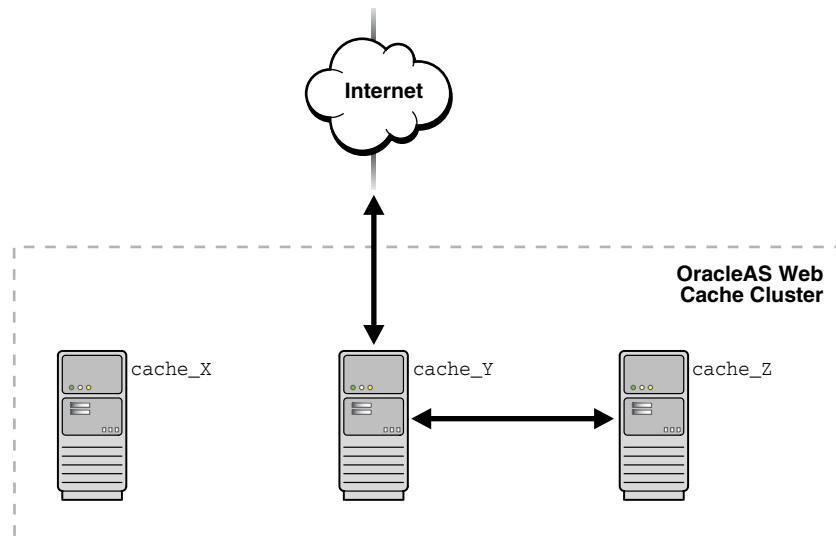
* If the requested content is not stored in the owner's cache, the owner, cache_Z, sends the request to the application Web server. The application Web server returns the requested content to the owner. The owner caches

the content and sends the requested content to the cluster member (cache_ Y) that originally received the request. Cache_Y returns the requested content to the client and stores the content as on-demand content, as shown in the following figure:
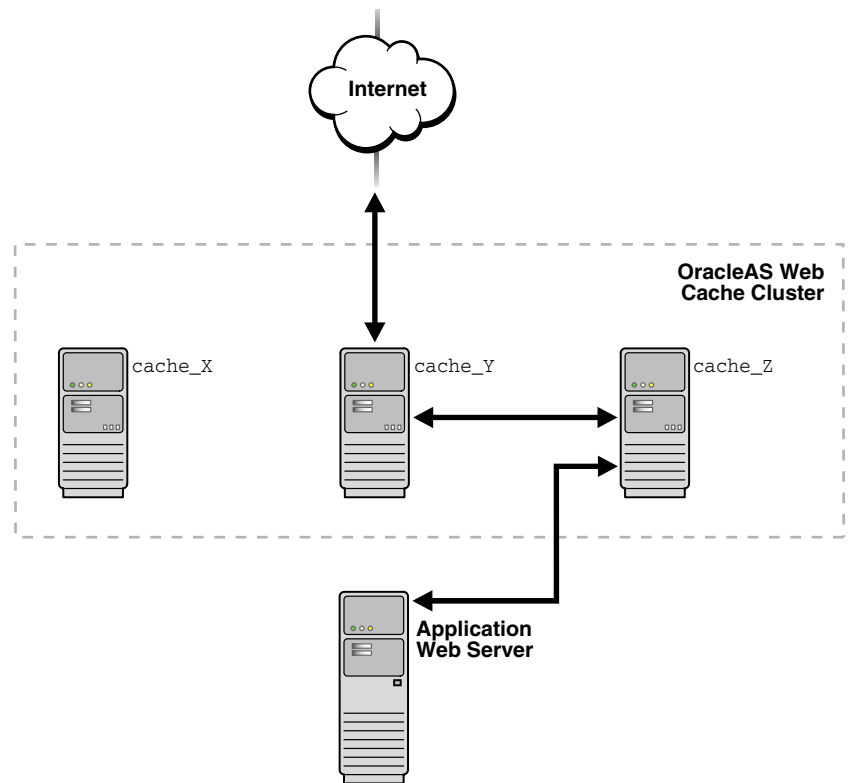
- If cache_Y cannot contact the owner (because of network problems or because the server for the owner has failed), it sends the request to the application Web server. The application Web server returns the requested content to cache_Y. Then, cache_ Y returns the requested content to the client and stores the content as on-demand content, as shown in the following figure:
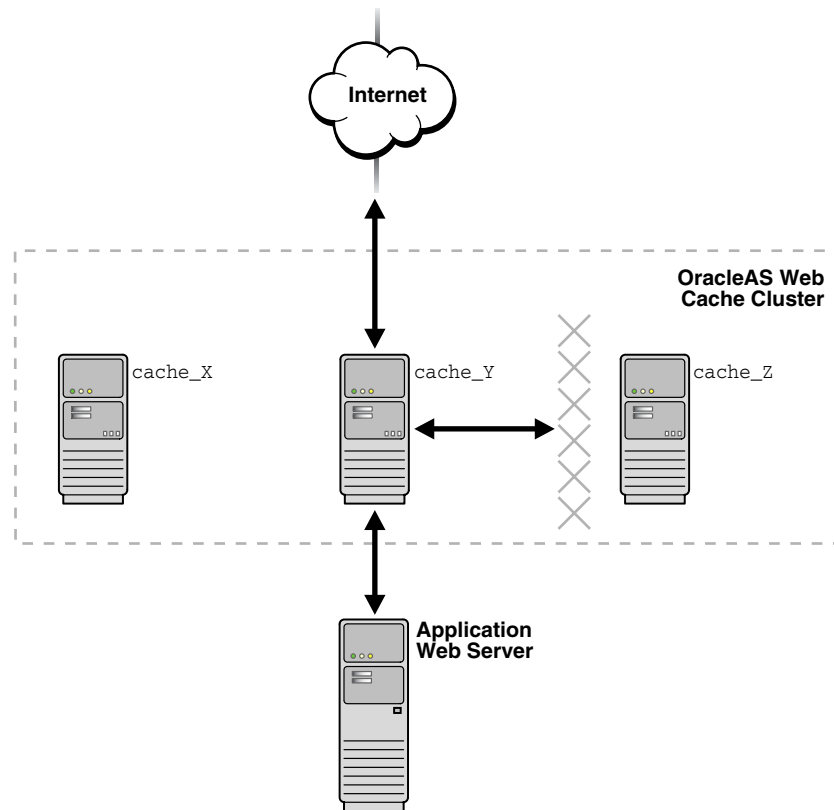
When you add a member to or remove a member from the cache cluster, OracleAS Web Cache uses the information about capacity to recalculate which cluster member owns which objects. If the ownership of an object changes and the object is currently cached, OracleAS Web Cache designates the object as on-demand content, rather than owned content, for that cache. The object is not removed from the previous owner cache, nor is it moved to the new owner cache. The object is not cached in the new owner cache until another request for the object is received.

Failure Detection and Failover

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

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

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

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

> **See Also:** Chapter 10 for information about configuring a cache cluster, including specifying a failover threshold

4

OracleAS Web Cache Security

The ability to control user access to Web content and to protect your site against people breaking into your system is critical. This chapter describes the architecture and configuration of security for OracleAS Web Cache:

- About OracleAS Web Cache Security
- Configuring OracleAS Web Cache Security

> **See Also:**
>
> - The *Oracle Application Server Security Guide* for an overview of Oracle Application Server security and its core functionality
> - The *Oracle Identity Management Concepts and Deployment Planning Guide* for guidance for administrators of the Oracle security infrastructure

About OracleAS Web Cache Security

This section describes the OracleAS Web Cache security model. It contains the following topics:

- OracleAS Web Cache Security Model
- Classes of Users and Their Privileges
- Resources Protected
- Authorization and Access Enforcement
- Leveraging Oracle Identity Management Infrastructure

OracleAS Web Cache Security Model

OracleAS Web Cache provides the following security-related features:

- Restricted Administration
- Secure Sockets Layer (SSL)
- SSL Acceleration

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

Restricted Administration

OracleAS Web Cache restricts administration with the following features:

- Password authentication for administration and invalidation operations

- Control over which ports are used for administration and invalidation operations

- IP and subnet administration restrictions

Secure Sockets Layer (SSL)

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, OracleAS Web Cache is able to cache pages for **HTTPS protocol** requests.

As shown in Figure 4–1, you can configure OracleAS Web Cache to receive HTTPS client requests and send HTTPS requests to origin servers. Typically, HTTP requests to origin servers use port 80, and HTTPS requests use port 443.

Figure 4–1   SSL for Secure Connections
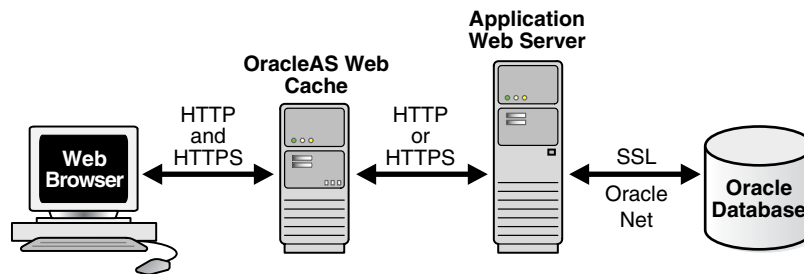
When sending requests to origin servers, note that HTTPS traffic can be processor intensive. If traffic from OracleAS Web Cache to an origin server must travel over the open Internet, configure OracleAS Web Cache to send HTTPS requests to the origin servers. If traffic only travels through a LAN in a data center, the traffic can be sent with HTTP so as to reduce the load on the origin servers.

OracleAS Web Cache supports both server-side and client-side certificates.

> **Limitations:** For this release, HTTPS support in OracleAS Web Cache does not provide authentication or access control natively. However, you can deploy OracleAS Web Cache with applications that require Oracle Application Server Single Sign-On. See "Routing Single Sign-On Server Requests" on page 5-14 for further information about configuring OracleAS Web Cache with Oracle Application Server Single Sign-On.

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 certificate you use in OracleAS Web Cache 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 Wallet Manager automatically installs with trusted certificates from VeriSign, RSA, Entrust, and GTE CyberTrust.

Certificate A certificate is a digital data record used for authenticating network entities such as a server or a client. It 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.

OracleAS Web Cache supports the following:

- **Server-side certificates:** A server-side certificate is a method for verifying the identity of the contacted server. It binds information about the server to the server's public key and must be signed by a trusted CA.

  For server-side certificates, OracleAS Web Cache sends the server certificate to the client browser during the SSL handshake, then processes the request for the object. If the requested object is not stored in the cache, the cache forwards the request to the application Web server, a peer cache (in a cluster), or a subordinate cache (in a hierarchy).

  You configure the caches to listen for HTTPS requests.

- **Client-side certificates:** A client-side certificate is a method for verifying the identity of the client. It binds information about the client user to the user's public key and must be digitally signed by a trusted CA.

  For client-side certificates, the client browser sends the certificate to the cache during the SSL handshake, then the cache processes the request for the object. If the requested object is not stored in the cache, the cache forwards the request to the application Web server, a peer cache (in a cluster), or another cache (in a hierarchy). To transfer information about the client-side certificate to another cache or to the application Web server, OracleAS Web Cache adds HTTP headers to the request. These headers begin with the string `SSL-Client-Cert`.

  You configure the caches to listen for HTTPS requests and to require client-side certificates.

In addition, depending on your deployment, you configure caches to accept the certificate information in HTTP headers from peer caches or from any entities (such as a provider or remote cache) or to not accept the certificate information in headers.

Note the following about server-side certificates:

- OracleAS Web Cache installs a default certificate that should only be used for testing purposes. OracleAS Web Cache prompts you to obtain a licensed certificate from the CA.

- One server-side certificate is required for each unique site configuration. HTTPS does not support multiple virtual hosts on a single port. For example, an environment with 20 site IP address and port number configurations requires 20 separate certificates.

Note the following about client-side certificates:

- Client-side certificates are not required for HTTPS requests. They are generally used when PKI-based user authentication is needed, such as in finance, government, or military applications.

- You can specify that an entire site require client-side certificates.

- If client-side certificates are required, but not provided by the client, OracleAS Web Cache returns an error: `403: Forbidden.`

- OracleAS Web Cache supports the use of client-side certificates with Oracle HTTP Server only.

- OracleAS Web Cache does not support client-side certificates with a distributed cache hierarchy because the security of the certificates cannot be guaranteed.

- In an ESI cache hierarchy, a provider cache must be able to accept the client-side certificate information in HTTP headers from the subscriber cache. However, with this configuration, the provider caches could inadvertently accept the certificate information in a header from a bogus entity. To prevent this, you must secure the provider caches, by methods such as installing them behind a firewall.

- Although the Oracle HTTP Server supports OpenSSL certificate revocation lists, OracleAS Web Cache does not. If you use client-side certificates, you must modify your application to check if the client-side certificate has been revoked. You can do this using a CGI script or servlet.

- Oracle Application Server does not support Microsoft Server Gated Cryptography Certificates (SGC) or VeriSign Global Server IDs. This cryptography enables export version browsers to transparently upgrade to strong 128-bit encryption from weaker 40-bit encryption when communicating with an application server. Without this cryptography, browsers with the weaker 40-bit encryption cannot negotiate a secure connection to Oracle Application Server. A future release of Oracle Application Server may support SGC and Global Server IDs.

> **See Also:** "Task 7: (Optional) Require Client-Side Certificates" on page 9-6

Wallet A wallet is a transparent repository 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 Oracle Wallet Manager to manage security credentials on the OracleAS 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 configure HTTPS for OracleAS Web Cache, create a wallet on the OracleAS Web Cache server for each supported site. You specify the location of the wallet for each of the OracleAS Web Cache HTTPS listening and operations ports (to support incoming HTTPS requests), and the origin server (to support outgoing HTTPS requests). You can share one wallet, or you can create separate wallets. If you use the same wallet, keep in mind that it can support only one server-side certificate.

Note that OracleAS Web Cache installs a default wallet with a default certificate, but this wallet should only be used for testing purposes, not in production environments. The SSL connection is not considered secure when using the default wallet. In a production environment, create a new wallet and create a new certificate or import a trusted certificate into the wallet.

> **See Also:**
>
> - *Oracle Application Server Administrator's Guide* for further information about Oracle Wallet Manager
>
> - Chapter 9 for configuration instructions

How SSL Works To describe how SSL works in an HTTPS connection, the word client is used to describe either a browser or OracleAS Web Cache, and the word server is used to describe either OracleAS Web Cache or an origin server. For example, when a browser is the client, the server can be OracleAS Web Cache or an origin server; when OracleAS Web Cache is the client, the server can be 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 to establish a secure connection.

An SSL handshake includes the following actions:

1. The client and server establish which cipher suites to use.

2. The server sends its certificate to the client, and the client verifies that the server's certificate was signed by a trusted CA.

3. Optionally, the server requests a client certificate and the client responds by sending the client certificate to the server. The server verifies that the client certificate was signed by a trusted CA.

4. 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.

> **See Also:**
>
> - Chapter 9 for configuration details
>
> - Chapter 17 for common SSL errors
>
> - Appendix E for troubleshooting tips

SSL Acceleration

In addition to offboard SSL acceleration solutions, Oracle Application Server now supports nCipher's BHAPI-compliant hardware for deployment on servers running OracleAS Web Cache and Oracle HTTP Server. When executed in software, SSL operations place a strain on server CPU resources, causing a reduction in throughput and slower overall performance. The nCipher hardware offloads the SSL key exchange processing from a server's CPUs, increasing the number of concurrent SSL connections and improving response times for SSL-protected content.

> **See Also:**
>
> - "Deploying OracleAS Web Cache with SSL Acceleration Hardware" on page 5-8
>
> - http://www.ncipher.com for more information about nCipher

Classes of Users and Their Privileges

OracleAS Web Cache provides support for the administrator and invalidator roles. The administrator can perform the following tasks:

- Start, stop, and restart OracleAS Web Cache

- Change configuration settings

- Send invalidation requests

- Send statistics monitoring requests

Both the Oracle Application Server administrator, ias_admin, and the OracleAS Web Cache administrator, administrator, accounts can use the administrator role. In a **cache cluster**, the password for all **cluster members** must be the same. By default, the password is the password you supplied during installation for the administrator username.

The invalidator role is limited to sending invalidation requests. By default, the password is the password you supplied during installation for the administrator username. In a cache cluster, the password for all cluster members must be the same.

You can change the passwords for administrator role or the invalidator role in the Security page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Security**) or OracleAS Web Cache Manager (**Properties** > **Security**).

> **See Also:**
>
> - "Starting OracleAS Web Cache Manager" on page 6-11 for more information on the administrator role and its password.
>
> - "Invalidation from External Sources" on page 13-1 for an explanation of how the administrator and invalidator roles send invalidation requests
>
> - "Task 2: Modify Security Settings" on page 8-7 for instructions on changing the administrator and invalidator passwords

Resources Protected

By default, the user that performed the installation is the owner of OracleAS Web Cache files. These files are readable by user ID and group ID specified in the Security page of Application Server Control Console (**Web Cache Home** page >

Administration tab > **Properties** > **Web Cache** > **Security**) or the Process Identity page of OracleAS Web Cache Manager (**Properties** > **Process Identity**). By default, the user that performed the installation is the owner of OracleAS Web Cache processes.

If you change the process identity user, you must manually change the ownership of OracleAS Web Cache files and directories to the new user ID and group ID with the `chown` command.

> **See Also:**
>
> - Step 3e of "Task 2: Modify Security Settings" on page 8-7 for information about modifying the process identity setting
>
> - "Problem 8: Permission Denied Error" on page E-6 for information about resolving process identity issues

Authorization and Access Enforcement

The `mod_access` module of Oracle HTTP Server controls access to the URLs based on characteristics of a request, such as host name or IP address. OracleAS Web Cache does not restrict IP address restrictions on a URL basis. If you are using `mod_access` with OracleAS Web Cache, ensure that the protected resources are not cached either by not specifying a caching rule or by explicitly setting a caching rule not to cache the content.

To pass the client IP directly to the Oracle HTTP Server, configure the `Order` directive in the `httpd.conf` file.

> **See Also:** *Oracle HTTP Server Administrator's Guide*

Leveraging Oracle Identity Management Infrastructure

The Oracle Identity Management infrastructure centralizes management of security across the enterprise.

Oracle Application Server Single Sign-On Servers

For security reasons, you should not cache content from Oracle Application Server Single Sign-On servers.

Oracle Application Server Single Sign-On Partner Applications (mod_osso)

You can configure OracleAS Web Cache to cache content for Oracle HTTP Servers running Single Sign-On partner applications. By default, `mod_osso` protected pages are configured as non-cacheable with a `Surrogate-Control: no-store` response header.

To override `mod_osso` default behavior, set `OssoSendCacheHeaders` to `off` in the `httpd.conf` file. For example:

```
<Location /foo/>
OssoSendCacheHeaders off
</Location>
```

This example disables the setting by `mod_osso` of any cache headers for any URL that starts with `/foo`. For these URLs, the application is responsible for setting the cache control headers, including `Surrogate-Control` as appropriate.

If OracleAS Web Cache is load balancing requests for identical Single Sign-On partner applications, configure the Oracle HTTP Servers as a cluster so that together, the applications act as a single partner application. You can then configure OracleAS Web

Cache to perform stateless load balancing of requests to the servers. If the application mid-tier is not clustered, stateful load balancing is necessary.

> **See Also:**
>
> - "Routing Single Sign-On Server Requests" on page 5-14 for further information about deploying OracleAS Web Cache with Oracle Application Server Single Sign-On
>
> - "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25 for instructions about configuring origin server and stateless load balancing settings
>
> - "Bind a Session to an Origin Server" on page 8-39 for instructions on configuring stateful load balancing settings
>
> - *Oracle Application Server Single Sign-On Administrator's Guide* for further information about clustering `mod_osso` on the Oracle HTTP Servers

Configuring OracleAS Web Cache Security

> **See:** Chapter 9 for information about configuring OracleAS Web Cache for HTTPS requests

5

OracleAS Web Cache Topologies

The *Oracle Application Server Concepts* and *Oracle Application Server Installation Guide* provide an overview of the recommended topologies for Oracle Application Server. This chapter presents several detailed scenarios for deploying OracleAS Web Cache.

> **Note:** The topologies described in this chapter depict OracleAS Web Cache on a dedicated computer. However, you can deploy OracleAS Web Cache on the same computer as the Oracle Application Server.

This chapter contains these topics:

- Common OracleAS Web Cache Configuration
- Specialized Topologies
- Security Topologies

Common OracleAS Web Cache Configuration

Figure 5–1 on page 5-2 shows OracleAS Web Cache in a common Oracle Application Server configuration. A tier of OracleAS Web Cache servers cache content for a tier of application Web servers. The application Web servers `app1-host1` and `app1-host2` provide content for site `www.app1.company.com`, and `app2-host` provides content for `www.app2.company.com`. The two OracleAS Web Cache servers reside on dedicated, fast one or two-CPU computers. To increase the availability and capacity of a Web site, these servers are configured as either a **cache cluster** or a failover pair.

Oracle recommends a hardware **load balancer** to ping each OracleAS Web Cache server on a periodic basis to check the status of the cache. You should configure the load balancer with the same ping URL you configure for the auto-restart mechanism, as described in"Task 3: Configure Auto-Restart Settings" on page 8-10.

As a cache cluster, the two OracleAS Web Cache servers provide failure detection and failover. If an OracleAS 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 and masks any cache failure. OracleAS Web Cache maintains a virtual single cache of content despite a cache failure. The load balancer distributes the incoming requests among cache cluster members. The cache cluster members process the incoming requests. For requests that are not stored in the cache, OracleAS Web Cache distributes the requests to an application Web server respective to the site.

As a failover pair, both OracleAS Web Cache servers are configured to cache the same content. When both OracleAS 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.

Figure 5–1   Deploying OracleAS Web Cache In a Common Configuration
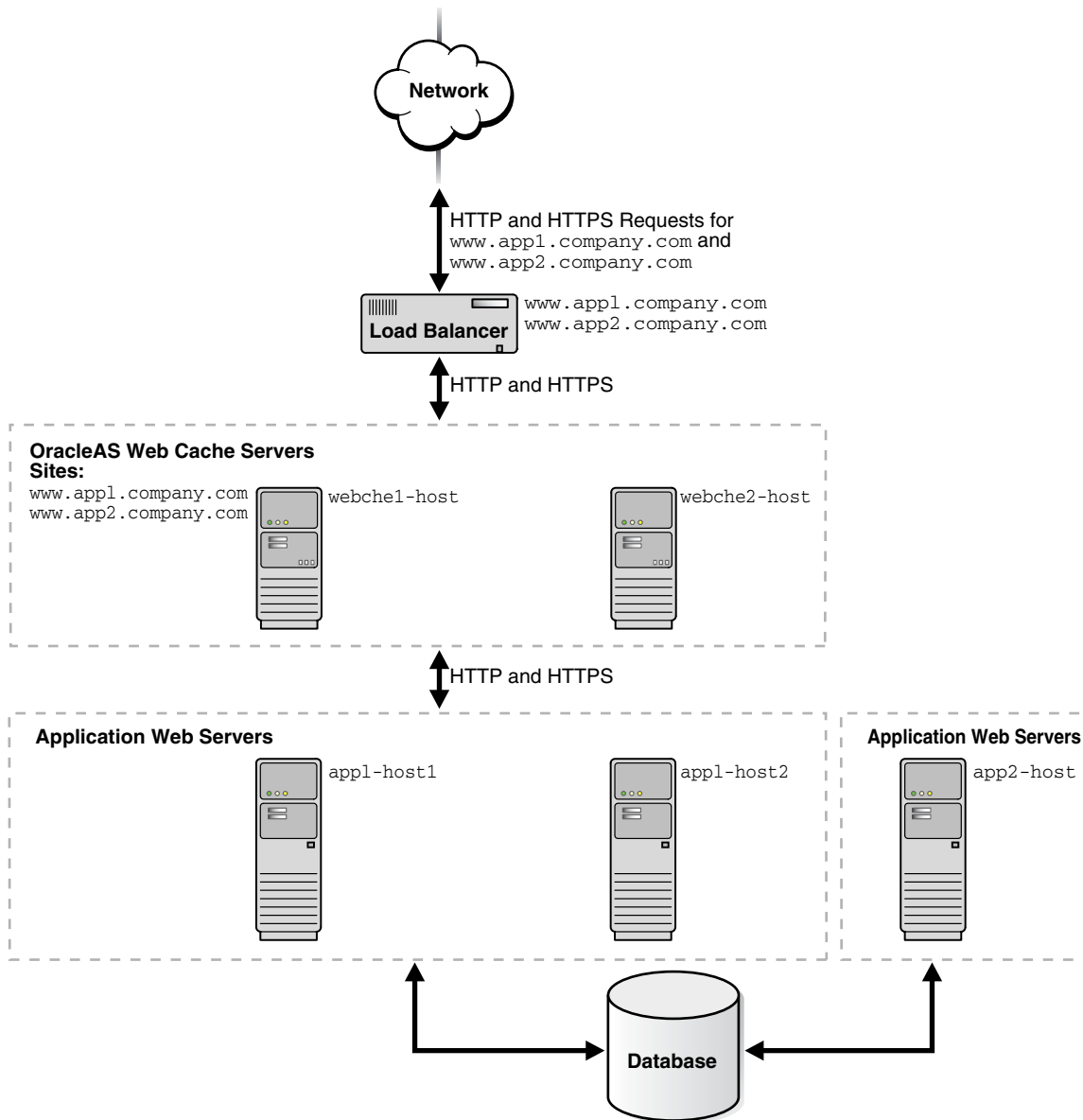
To configure this topology:

1. Register the IP address of the load balancer with `www.app1.company.com` and `www.app2.company.com`.

2. Configure the load balancer with OracleAS Web Cache server host names `webche1-host` and `webche2-host` and configure it to ping each cache server periodically to check the status of the cache.

3. If configuring a cache cluster, specify `webche1-host` and `webche2-host` as cluster members.

> **See Also:**   Chapter 10 for instructions on creating a cache cluster

4. Configure the OracleAS Web Cache servers with the following:

- Receive HTTP and HTTPS requests on designated listening ports

- Send HTTP and HTTPS requests to application Web servers `app1-host1`, `app1-host2`, and `app2-host` on designated listening ports

- Virtual host site definition for `www.app1.company.com` mapped to `app1-host1` and `app1-host2`

- Virtual site definition for `www.app2.company.com` mapped to `app2-host`

  See Also:

  - "Task 6: Configure OracleAS Web Cache with Listening Ports for Client Requests" on page 8-19 for instructions about configuring listening ports

  - "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25 for instructions about configuring origin server settings

  - "Task 10: Configure Web Site Settings" on page 8-27 for instructions about creating site definitions and site-to-server mappings

Specialized Topologies

This section describes the following specialized topologies:

- Deploying a Distributed Cache Hierarchy

- Deploying OracleAS Web Cache for High Availability Without a Hardware Load Balancer

- Deploying OracleAS Web Cache With a Layer 7 Switch

Deploying a Distributed Cache Hierarchy

Many Web sites have several data centers. For networks with a distributed topology, you can deploy OracleAS Web Cache at each of the data centers in a **distributed cache hierarchy**. Figure 5–2 on page 5-4 shows a distributed topology in which OracleAS Web Cache servers are distributed in offices in the United States and Japan. The application Web servers are located in the United States office, centralizing the data source to one geographic location. The **central caches** in the United States cache content for application Web servers `app1-host1`, `app2-host2`, and `app2-host`, and the **remote cache** in Japan caches content from the central caches.

Clients make requests to local DNS servers to resolve `www.app1.company.com` and `www.app2.company.com`. The local DNS servers are routed to the authoritative DNS server for the respective sites. The authoritative DNS server uses the IP address of the client to pick the closest OracleAS Web Cache server to satisfy the request. Then, it returns the IP address of the appropriate OracleAS Web Cache server to the client.

Figure 5–2   Deploying an OracleAS Web Cache Hierarchy
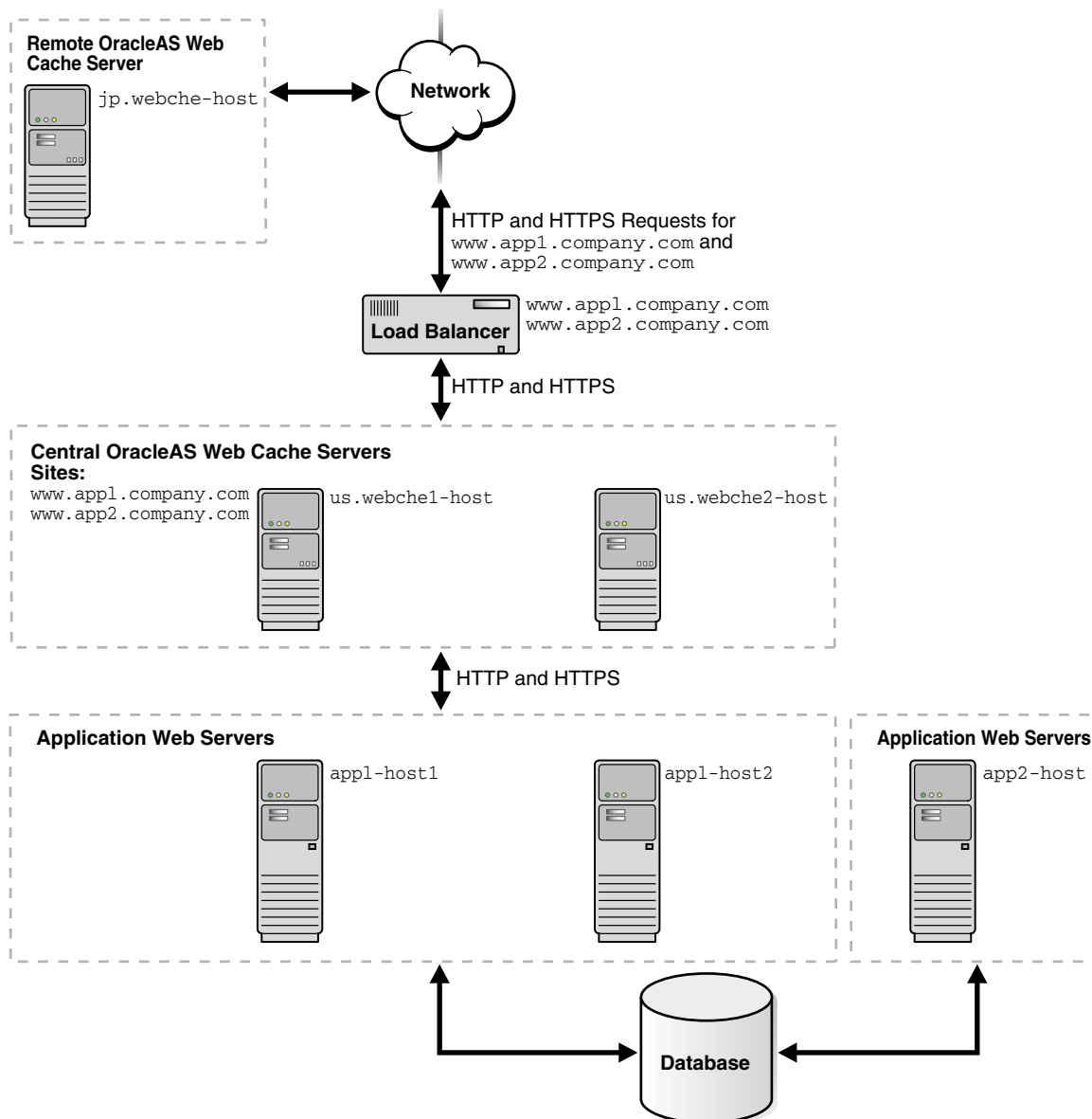
To configure this topology:

1.   Register the IP address of the load balancer with `www.app1.company.com` and `www.app2.company.com`.

2.   Configure the load balancer with OracleAS Web Cache server host names `us.webche1-host` and `us.webche2-host` and configure it to ping each cache server periodically to check the status of the cache.

3.   Configure OracleAS Web Cache servers `us.webche1-host` and `us.webche1-host` with the following:

   ■   Receive HTTP and HTTPS requests on designated listening ports

   ■   Send HTTP and HTTPS requests to application Web servers `app1-host1`, `app1-host2`, and `app2-host` on designated listening ports

- Virtual host site definition for `www.app1.company.com` mapped to `app1-host1` and `app1-host2`

- Virtual site definition for `www.app2.company.com` mapped to `app2-host`

4. Configure OracleAS Web Cache server `jp.webche-host` with the following:

- Receive HTTP and HTTPS requests on designated listening ports

- Send HTTP and HTTPS requests to application Web servers `us.webche1-host` and `us.webche2-host` on designated listening ports

- Virtual host site definition for `www.app1.company.com` mapped to `app1-host1` and `app1-host2`

- Virtual site definition for `www.app2.company.com` mapped to `app2-host`

   See Also:

   - Chapter 11 for more details about this configuration

   - "Task 6: Configure OracleAS Web Cache with Listening Ports for Client Requests" on page 8-19 for instructions about configuring listening ports

   - "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25 for instructions about configuring origin server settings

   - "Task 10: Configure Web Site Settings" on page 8-27 for instructions about creating site definitions and site-to-server mappings

Deploying OracleAS Web Cache for High Availability Without a Hardware Load Balancer

You can make OracleAS Web Cache highly available without a hardware load balancer by configuring:

- OracleAS Web Cache solely as a software load balancer of HTTP traffic or **reverse proxy** to origin servers

   With this option, you configure one or more caches solely to provide load balancing or reverse proxy support.

- Operating system load balancing capabilities

   With this option, you configure the operating system to load-balance incoming requests across multiple caches. When the operating system detects a failure of one of the caches, automatic IP takeover is used to distribute the load to the remaining caches in the cluster configuration. This feature is supported on many operating systems, including Linux, Windows 2000 Advanced Server, Windows 2000 Datacenter Server, and Windows 2003 (all editions).

   See Also:

   - "OracleAS Web Cache Solely as a Software Load Balancer or Reverse Proxy" on page 8-42

   - "Operating System Load Balancing Support" on page 8-45

Deploying OracleAS Web Cache With a Layer 7 Switch

Many Web sites contain cacheable public content and non-cacheable, transactional or protected content. For these Web sites, you can use OracleAS Web Cache servers to cache content for only the portions of the Web site with the cacheable content.

Figure 5–3 shows a **Layer 7 (L7) switch** passing catalog requests to OracleAS Web Cache servers `webche1-host` and `webche2-host` and order entry requests to application Web server `app1-host1`. 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 5–3   Accelerating Portions of a Web Site with an Layer 7 Switch
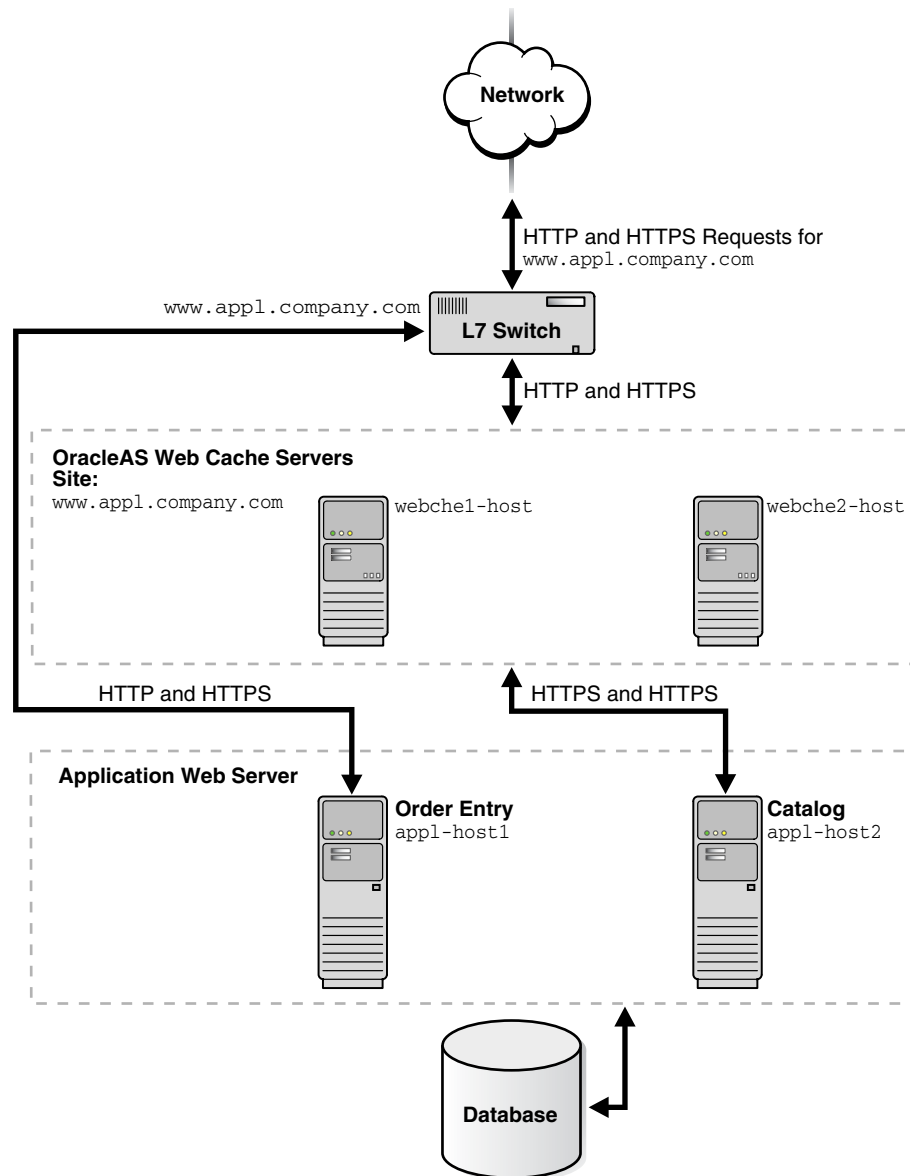
To configure this topology:

1. Register the IP address of the L7 switch with `www.app1.company.com`.

2. Configure the L7 switch with OracleAS Web Cache server host names `webche1-host` and `webche2-host`.

3. Configure the OracleAS Web Cache servers with the following:

- Receive HTTP and HTTPS requests on designated listening ports

- Send HTTP and HTTPS requests to application Web servers `app1-host1` and `app1-host2` on designated listening ports

- Virtual host site definition for `www.app1.company.com` mapped to `app1-host1` and `app1-host2`

See Also:

- "Task 6: Configure OracleAS Web Cache with Listening Ports for Client Requests" on page 8-19 for instructions about configuring listening ports

- "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25 for instructions about configuring origin server settings

- "Task 10: Configure Web Site Settings" on page 8-27 for instructions about creating site definitions and site-to-server mappings

Security Topologies

This section describes the following topologies:

- With OracleAS Web Cache deployed inside or outside a firewall. "Deploying OracleAS Web Cache with Firewalls" on page 5-7 describes this scenario.

- With OracleAS Web Cache listening for incoming requests on two ports, one for HTTPS requests and one for HTTP requests. You can configure a load balancer to pass requests to the appropriate listening port. In addition, you can configure OracleAS Web Cache with SSL acceleration hardware to off-load the SSL operations. "Deploying OracleAS Web Cache with SSL Acceleration Hardware" on page 5-8 describes this scenario.

- With one OracleAS Web Cache server listening for HTTP requests and another OracleAS Web Cache server listening for HTTPS requests. You can configure a load balancer to pass requests to the appropriate OracleAS Web Cache server. "Routing HTTPS Requests to a Dedicated Cache" on page 5-10 describes this scenario.

- With OracleAS Web Cache listening only for HTTP requests. All HTTPS requests are routed to the application Web server. You can configure a load balancer to pass HTTPS requests directly to the application Web server. "Routing HTTPS Requests Around OracleAS Web Cache" on page 5-12 describes this scenario.

- With OracleAS Web Cache caching content for Oracle HTTP Servers running Oracle Application Server Single Sign-On partner applications and load balancing the redirected requests from the partner applications to the Single Sign-On servers. "Routing Single Sign-On Server Requests" on page 5-14 describes these scenarios.

See also: Chapter 9 for configuration details

Deploying OracleAS Web Cache with Firewalls

You can deploy OracleAS Web Cache inside or outside a firewall. Deploying OracleAS 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. When deploying OracleAS Web Cache outside a firewall, the throughput burden is placed on OracleAS Web Cache rather than the firewall. The

firewall receives only requests that must go to the application Web servers. This topology requires securing OracleAS Web Cache from intruders.

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

Deploying OracleAS Web Cache with SSL Acceleration Hardware

You can configure OracleAS Web Cache to receive both HTTP and HTTPS requests. You can off-load some of the strain of SSL operations on the OracleAS Web Cache CPUs by using SSL acceleration hardware.

Figure 5–4 on page 5-9 shows OracleAS Web Cache servers `webche1-host` and `webche2-host` deployed with SSL acceleration hardware. Both servers receive HTTP and HTTP requests, with the HTTPS requests being processed by the SSL acceleration hardware. Both OracleAS Web Cache servers send origin server requests to `app1-host1` and `app1-host2`.

> **Note:** Oracle Application Server supports nCipher's BHAPI-compliant hardware for deployment on OracleAS Web Cache servers. See `http://www.ncipher.com` for more information about nCipher.

Figure 5–4   Routing HTTPS Requests to SSL Acceleration Hardware
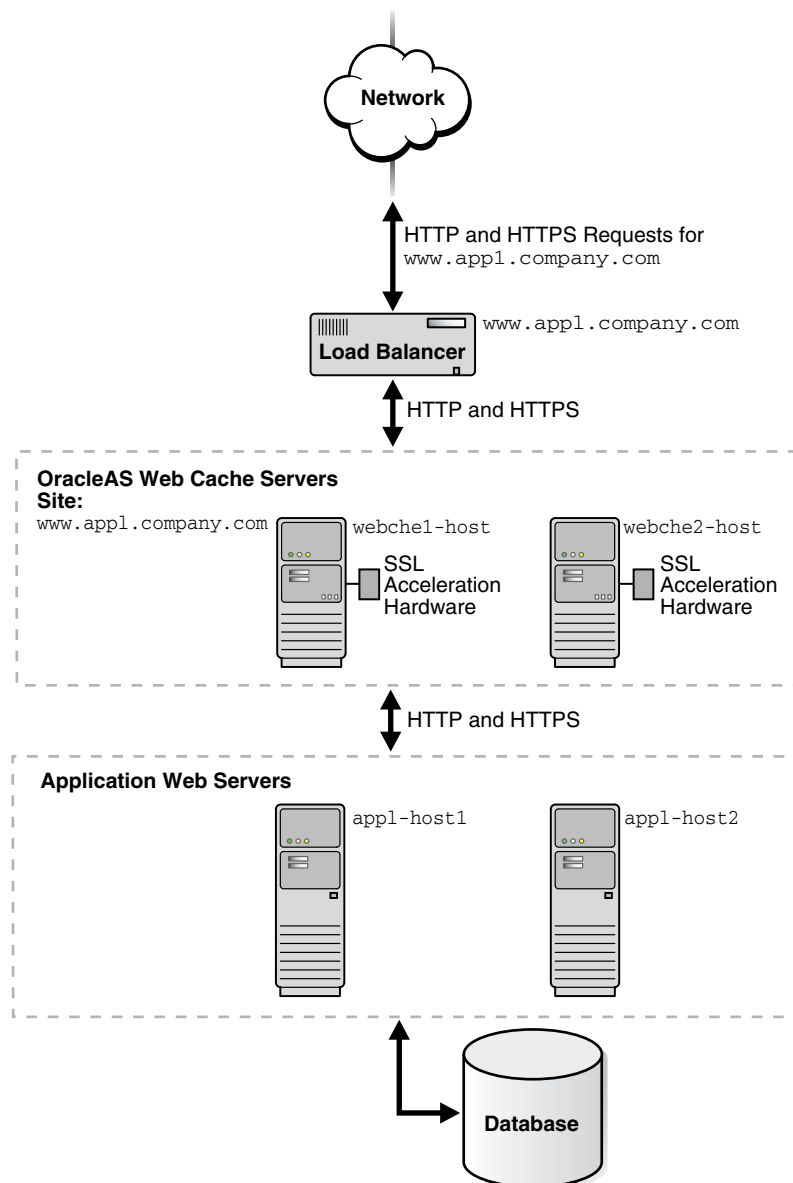
To configure this topology:

1. Ensure that the load balancer supports sticky sessions so that HTTPS requests for a given user session are bound to the same OracleAS Web Cache server.

2. Register the IP address of the load balancer with `www.app1.company.com`.

3. Configure the load balancer with OracleAS Web Cache server host names `webche1-host` and `webche2-host`. Configure the load balancer to send HTTP and HTTPS requests to `webche1-host` and `webche2-host` and to ping each cache server periodically to check the status of the cache.

4. Configure the OracleAS Web Cache servers with the following:

    ■   Receive HTTP and HTTPS requests on designated listening ports.

    ■   Send HTTP and HTTPS requests to application Web servers `app1-host1` and `app1-host2` on designated listening ports.

- Virtual host site definition for `www.app1.company.com`, with HTTP and HTTPS ports, mapped to `app1-host1` and `app1-host2`.

- If using client-side certificates, require client-side certificates for the HTTPS listening ports. In addition, for a cache cluster, allow cluster members to pass certificate information in headers to other cluster members and disallow acceptance of such headers from all other entities.

Note: In a cache cluster, you *must* disallow acceptance of headers containing client-side certificate information to prevent receipt of certificate information in a header from any entity other than a peer cluster member. See "Task 7: (Optional) Require Client-Side Certificates" on page 9-6 for more information.

See Also:

- Chapter 9 for detailed instructions for configuring HTTPS

- "Task 6: Configure OracleAS Web Cache with Listening Ports for Client Requests" on page 8-19 for instructions about configuring listening ports

- "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25 for instructions about configuring origin server settings

- "Task 10: Configure Web Site Settings" on page 8-27 for instructions about creating site definitions and site-to-server mappings

Routing HTTPS Requests to a Dedicated Cache

You can configure one OracleAS Web Cache server to listen for HTTP requests and another OracleAS Web Cache server to listen for HTTPS requests.

Figure 5–5 on page 5-11 shows two OracleAS Web Cache servers receiving requests. HTTP requests are served from server `webche1-host` and HTTPS requests are served from server `webche2-host`. Both OracleAS Web Cache servers send origin server requests to `app1-host1` and `app1-host2`.

Figure 5–5   Routing HTTPS Requests To a Dedicated OracleAS Web Cache Server
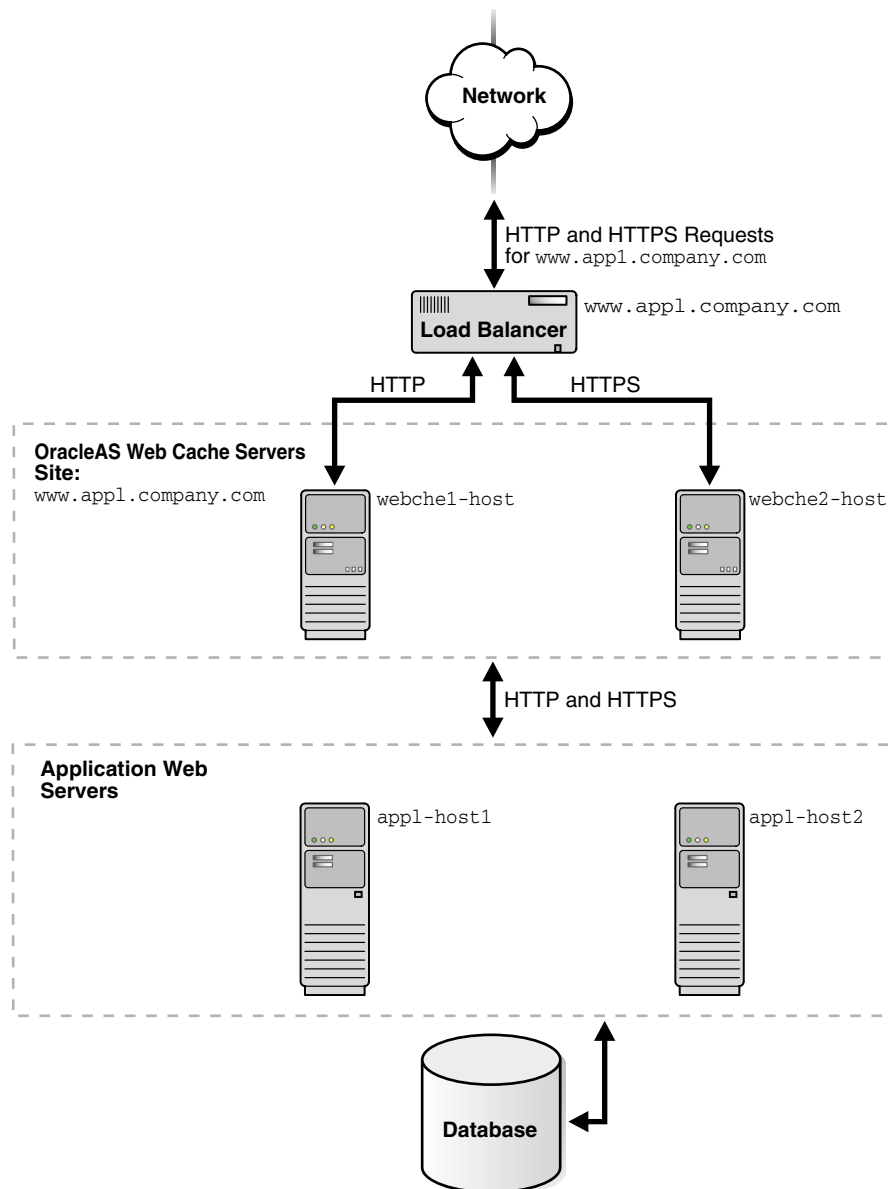
To configure this topology:

1. Register the IP address of the load balancer with `www.app1.company.com`.

2. Configure the load balancer with OracleAS Web Cache server host names `webche1-host` and `webche2-host`. Configure the load balancer to send HTTP requests to `webche1-host` and HTTPS requests to `webche2-host` and to ping each cache server periodically to check the status of the cache.

3. Configure OracleAS Web Cache server `webche1-host` with the following:

    ■   Receive HTTP requests on a designated HTTP listening port

    ■   Send HTTP requests to application Web servers `app1-host1` and `app1-host2` on designated HTTP listening ports

    ■   Virtual host site definition for `www.app1.company.com`, with an HTTP port, mapped to `app1-host1` and `app1-host2`

4. Configure OracleAS Web Cache server `webche2-host` with the following:

- Receive HTTPS requests on a designated HTTPS listening port.

- Send HTTP requests to application Web servers `app1-host1` and `app1-host2` on designated HTTP listening ports.

- Virtual host site definition for `www.app1.company.com`, with an HTTPS port, mapped to `app1-host1` and `app1-host2`.

- If using client-side certificates, require client-side certificates for the HTTPS listening ports. In addition, for a cache cluster, allow cluster members to pass certificate information in headers to other cluster members and disallow acceptance of such headers from all other entities.

Note: In a cache cluster, you *must* disallow acceptance of headers containing client-side certificate information to prevent receipt of certificate information in a header from any entity other than a peer cluster member. See "Task 7: (Optional) Require Client-Side Certificates" on page 9-6 for more information.

See Also:

- Chapter 9 for detailed instructions for configuring HTTPS

- "Task 6: Configure OracleAS Web Cache with Listening Ports for Client Requests" on page 8-19 for instructions about configuring listening ports

- "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25 for instructions about configuring origin server settings

- "Task 10: Configure Web Site Settings" on page 8-27 for instructions about creating site definitions and site-to-server mappings

Routing HTTPS Requests Around OracleAS Web Cache

For many applications, HTTPS is required for secure transactions that should not be cached. For example, checkout 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 OracleAS Web Cache, and pass all HTTPS requests for secure pages directly to a particular application Web server.

Figure 5–6 on page 5-13 shows a load balancer passing HTTP requests to OracleAS Web Cache server `webche1-host` and `webche2-host` and HTTPS requests to application Web server `app1-host2`. Note that HTTPS requests could also be passed to `app1-host1`.

Figure 5–6   Routing HTTPS Requests To an Application Web Server
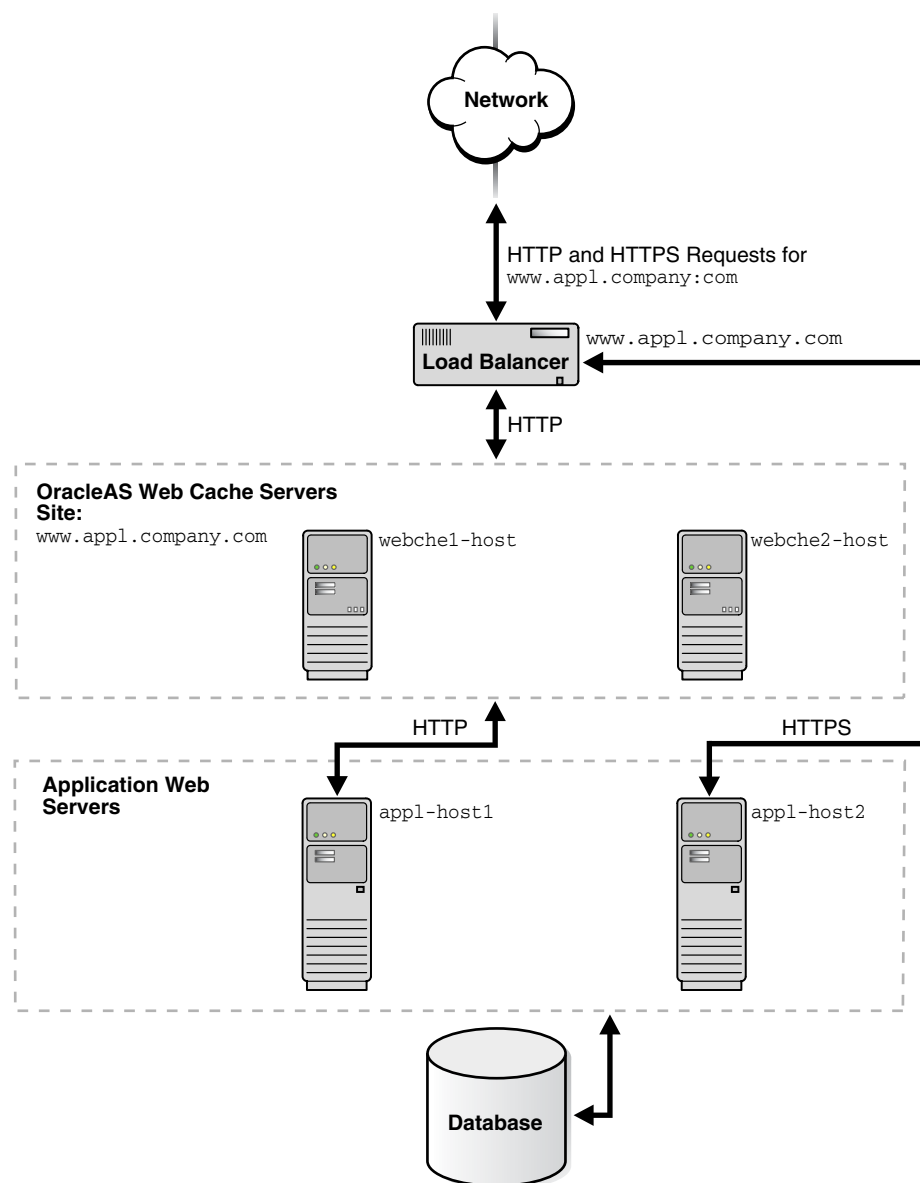
To configure this topology:

1. Register the IP address of the load balancer with `www.app1.company.com`.

2. Configure the load balancer with OracleAS Web Cache server host names `webche1-host` and `webche1-host` and application Web server host name `app1-host2`.

3. Configure the OracleAS Web Cache servers with the following:

   ■ Receive HTTP requests on a designated HTTP listening port

   ■ Send HTTP requests to application Web servers `app1-host1` on an HTTP listening port

   ■ Virtual host site definition for `www.app1.company.com`, with an HTTP port, mapped to `app1-host1`

> **See Also:**
>
> - "Task 6: Configure OracleAS Web Cache with Listening Ports for Client Requests" on page 8-19 for instructions about configuring listening ports
>
> - "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25 for instructions about configuring origin server settings
>
> - "Task 10: Configure Web Site Settings" on page 8-27 for instructions about creating site definitions and site-to-server mappings

Routing Single Sign-On Server Requests

Because login and logout requests to Single Sign-On servers are both secure and time sensitive, you cannot cache content from the Single Sign-On servers. You can configure OracleAS Web Cache as a software load balancer in front of multiple Single Sign-On mid-tiers, similar to a hardware-based load balancer approach. You can also configure OracleAS Web Cache to cache content for Oracle HTTP Servers running Oracle Application Server Single Sign-On partner applications. By default, `mod_osso` protected pages are marked as non-cacheable to ensure that requests for these pages are redirected through Oracle Application Server Single Sign-On servers.

Figure 5–7 on page 5-15 shows OracleAS Web Cache caching content for the Oracle HTTP Servers `app1-host` and `app1-host2` and a load balancer routing the requests from the partner applications to the Single Sign-On servers `sso-host1` and `sso-host2`. No content from the Single Sign-On servers is cached. This topology requires that the Oracle HTTP Servers be clustered so that OracleAS Web Cache can send requests to either partner application.

Figure 5–7   Routing Single Sign-On Server Requests to the Single Sign-On Mid-Tier Around OracleAS Web Cache
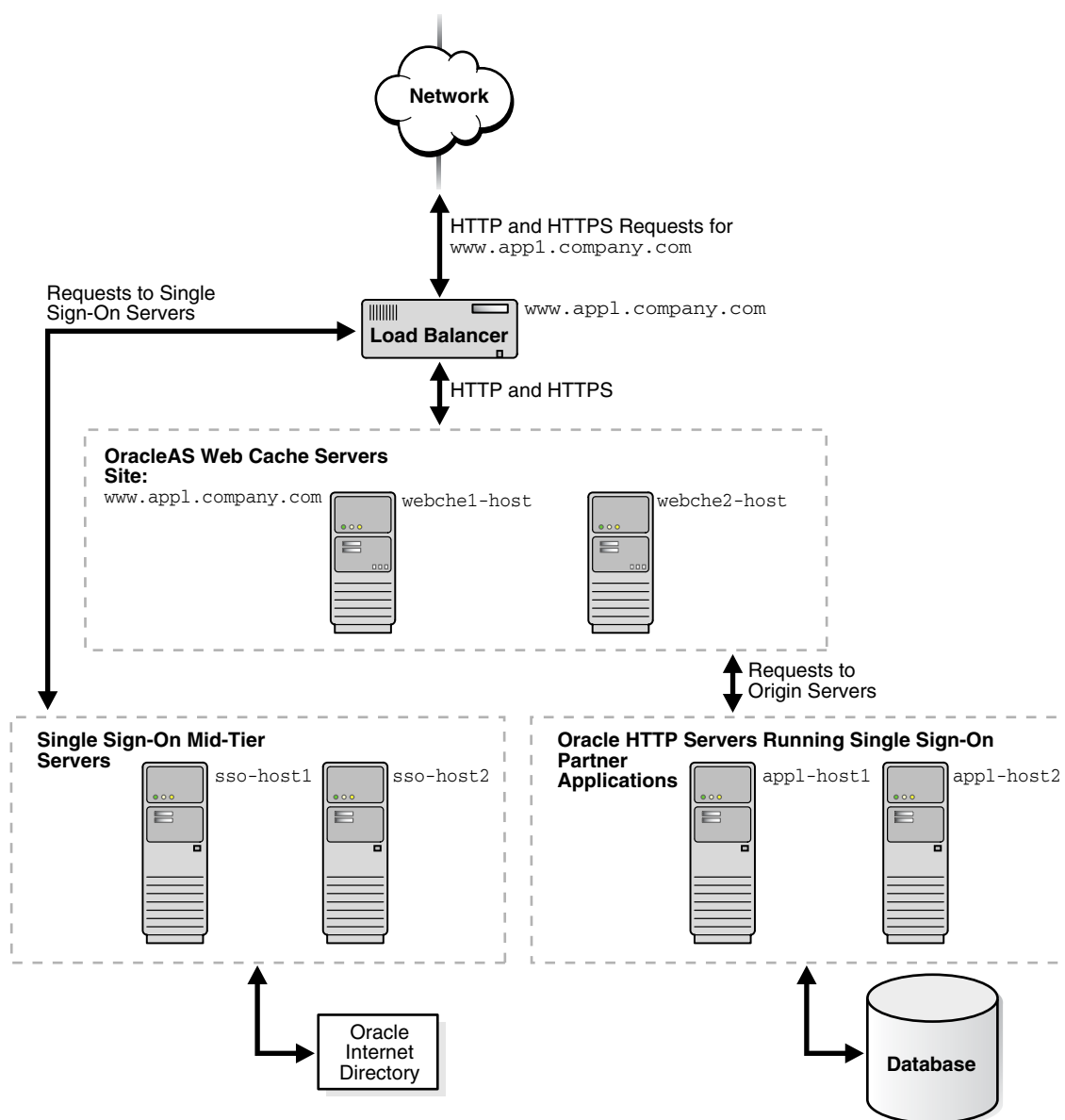
To configure this topology:

1. Ensure that the load balancer supports sticky sessions so that HTTPS requests for a given user session are bound to the same OracleAS Web Cache server.

2. Register the IP address of the load balancer with `www.app1.company.com`.

3. Configure the load balancer with OracleAS Web Cache server host names `webche1-host` and `webche2-host`. Configure the load balancer to send HTTP and HTTPS requests to `webche1-host` and `webche2-host`.

4. Configure the OracleAS Web Cache servers with the following:

    ■   Receive HTTP and HTTPS requests on designated listening ports.

    ■   Stateless load balancing of HTTP and HTTPS requests to clustered Oracle HTTP Servers `app1-host1` and `app1-host2` on designated listening ports.

- Virtual host site definition for `www.app1.company.com`, with HTTP and HTTPS ports, mapped to `app1-host1` and `app1-host2`.

- If using client-side certificates, require client-side certificates for the HTTPS listening ports. In addition, for a cache cluster, allow cluster members to pass certificate information in headers to other cluster members and disallow acceptance of such headers from all other entities.

> **Note:** In a cache cluster, you *must* disallow acceptance of headers containing client-side certificate information to prevent receipt of certificate information in a header from any entity other than a peer cluster member. See "Task 4: Configure HTTPS Port and Wallet Location for the Origin Server" on page 9-4 for more information.

See Also:

- "Leveraging Oracle Identity Management Infrastructure" on page 4-7 for further information about using OracleAS Web Cache with Oracle Application Server Single Sign-On servers and Single Sign-On partner applications

- "Task 6: Configure OracleAS Web Cache with Listening Ports for Client Requests" on page 8-19 for instructions about configuring listening ports

- "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25 for instructions about configuring origin server and stateless load balancing settings

- "Task 10: Configure Web Site Settings" on page 8-27 for instructions about creating site definitions and site-to-server mappings

- *Oracle Application Server Single Sign-On Administrator's Guide* for information about clustering `mod_osso` on the Oracle HTTP Servers

6

Introduction to Administration Tools

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

This chapter contains these topics:

- Overview of Tools for Managing OracleAS Web Cache within Oracle Application Server
- Overview of Tools for Standalone Configurations
- Understanding the OracleAS Web Cache Configuration Files
- Configuration and Administration Tasks at a Glance
- Script for Setting File Permissions on UNIX

Overview of Tools for Managing OracleAS Web Cache within Oracle Application Server

Oracle offers the following options for managing OracleAS Web Cache within an Oracle Application Server installation:

- Managing OracleAS Web Cache with Oracle Enterprise Manager 10g Application Server Control Console
- Gathering Historical Metrics Data for OracleAS Web Cache with Oracle Enterprise Manager 10g Grid Control
- Managing Processes with Oracle Process Manager and Notification (OPMN)

> **See Also:**
>
> - *Oracle Application Server Administrator's Guide* for an introduction to using Oracle Enterprise Manager 10*g* with Oracle Application Server
> - "Starting and Stopping OracleAS Web Cache In an Oracle Application Server Configuration" on page 7-1

Managing OracleAS Web Cache with Oracle Enterprise Manager 10*g* Application Server Control Console

Oracle Enterprise Manager 10*g* Application Server Control Console (Application Server Control Console) enables you to manage Oracle Application Server components, including OracleAS Web Cache.

Application Server Control Console enables you to manage an Oracle Application Server instance. From the Application Server Control Console, you can configure OracleAS Web Cache, monitor cache and origin server status, and perform operational tasks, such as starting and stopping the cache, invalidate content, propagate configuration among cache cluster members, and manage log files.

Application Server Control is installed with every instance of Oracle Application Server. As a result, you can immediately begin managing your application server and its components from your Web browser.

Accessing the OracleAS Web Cache Pages within Application Server Control Console

To view the OracleAS Web Cache pages:

1. Display the Oracle Application Server Welcome Page by entering the following URL in your Web browser:

   `http://hostname.domain:port`

   For example, if you installed Oracle Application Server on a host called `sys42`, you would enter the following address in your Web browser:

   `http://sys42.acme.com:7777`

   > **Note:** The default port for Oracle HTTP Server (and, as a result, the Welcome page) is described in the text file (`setupinfo.txt`) that is generated and displayed at the end of the Oracle Application Server installation. This file is located in `$ORACLE_HOME/install` on UNIX and `ORACLE_HOME\install` on Windows.

2. Click **Log on to the Oracle Enterprise Manager 10g** Application Server Control Console.

   Enterprise Manager displays the administrator logon dialog box.

3. Enter the Oracle Application Server administrator user name and password and click **OK**.

   The user name for the administrator user is `ias_admin`. The password is the one you supplied during the installation of Oracle Application Server.

4. From the **System Components** table in the Application Server home page, click Web Cache.

   The Web Cache Home page (Figure 6–1 on page 6-3) appears.

Table 6–1 describes the main OracleAS Web Cache pages in the Application Server Control Console.

Table 6–1   Application Server Control Console Tabbed Pages for Managing OracleAS Web Cache

Enterprise Manager Web Cache Page	Description
Web Cache Home page	Use this home page to monitor OracleAS Web Cache and its activity.
	See Also: "Using the Web Cache Home Page" on page 6-3

Table 6–1 (Cont.) Application Server Control Console Tabbed Pages for Managing OracleAS Web Cache

Enterprise Manager Web Cache Page	Description
Web Cache Performance page	Use this page to monitor the efficiency of the cache.
	See Also: "Using the Web Cache Performance Page" on page 6-3
Web Cache Administration page	Use this page to perform configuration and administrative actions.
	See Also: "Using the Web Cache Administration Page" on page 6-4

Using the Web Cache Home Page

From the Web Cache Home page (Figure 6–1), you can start and stop the cache, monitor the overall performance of the server, and monitor the activity level of the cache.

Figure 6–1 Application Server Control Console: Web Cache Home Page

Using the Web Cache Performance Page

From the Web Cache Performance page (Figure 6–2 on page 6-4), you can monitor the efficiency of the cache. Specifically, you can view the high-level metrics you are interested in viewing: requests served in the **Requests** section, cache hits and misses in the **Hits and Misses** sections, compression performance in the **Compression** sections, and errors in the **Errors** section.

Figure 6–2   Application Server Control Console: Web Cache Performance Page
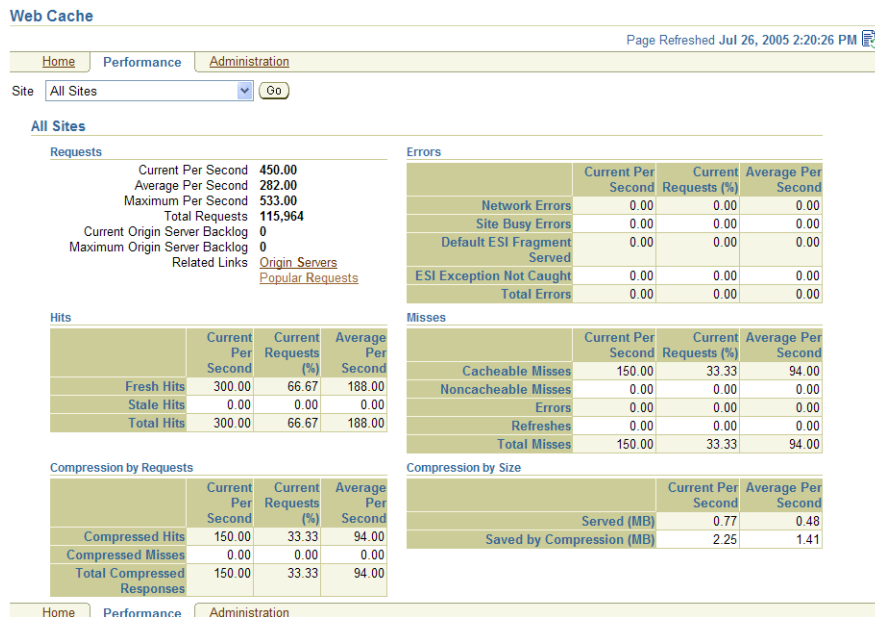
Using the Web Cache Administration Page

From the Web Cache Administration page (Figure 6–3), you can configure basic and advanced configuration functions to be stored in configuration file `webcache.xml`. The Web Cache Administration page contains links to property pages for editing this configuration file.

Figure 6–3   Application Server Control Console: Web Cache Administration Page
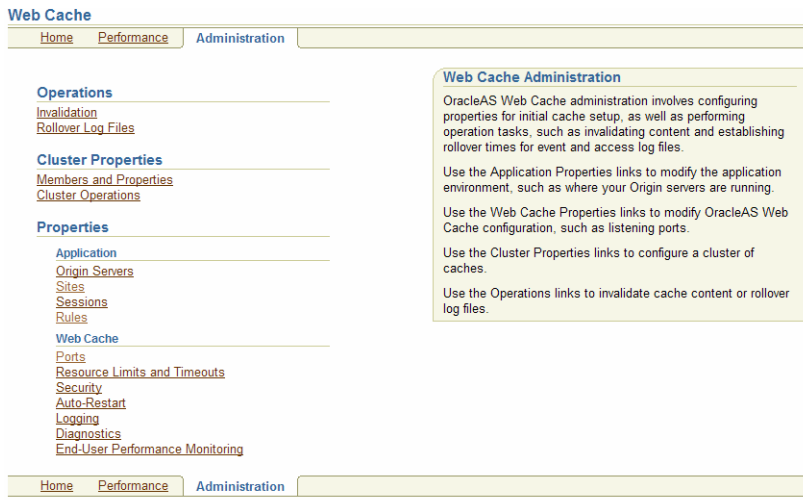
The page contains links for the following major categories:

- Operations: This category contains pages that enable you to:
  - Invalidate objects in the cache
  - Roll over log files
- Cluster Properties: This category contains pages that enable you to:

- – Configure a cache cluster
- – Check the status of the cache cluster members and perform operations on the current cache and other cluster members.
- Properties: Application: This category contains pages that enable you to:
  - – Specify origin servers
  - – Configure settings for sites, including site definitions and mappings to origin servers, session binding, and error pages
  - – Configure session definitions
  - – Configure caching rules
- Properties: Web Cache: This category contains pages that enable you to:
  - – Configure OracleAS Web Cache listening ports and operation ports for administration, statistics, and invalidation requests
  - – Configure security settings, such as usernames and passwords for administrators and invalidation users, and the process identity for OracleAS Web Cache processes
  - – Specify the size of the cache and the number of connections and network time-outs
  - – Specify settings for the auto-restart mechanism
  - – Configure event and access logging settings
  - – Configure diagnostic information
  - – Configure end-user performance monitoring

Using the Application Server Control Console Online Help

You can click **Help** at the top of the page to get more information. In most cases, the Help window displays a help topic about the current page. Click **Help Contents** in the Help window to browse the list of help topics or to search for a particular word or phrase.

Gathering Historical Metrics Data for OracleAS Web Cache with Oracle Enterprise Manager 10*g* Grid Control

Oracle Enterprise Manager 10*g* Grid Control is installed separately from the Oracle Enterprise Manager 10*g* installation CD-ROM. The Grid Control Console provides a wider view of your network so you can manage multiple Oracle Application Server instances. In addition, the Grid Control Console provides a robust feature set designed to help you manage all aspects of your enterprise, including your Oracle databases, hosts, listeners, and other components.

The Grid Control Console enables you to view historical metrics for multiple Oracle Application Server targets. For OracleAS Web Cache, use the Grid Control Console to view the overall status of OracleAS Web Cache and view OracleAS Web Cache performance metrics. Based on the metric data, you can make changes to the configuration with Grid Control Console.

Accessing the OracleAS Web Cache pages within Grid Control Console

To view the OracleAS Web Cache pages:

1. Log in to the Grid Control Console by entering the following URL in your Web browser:

   `http://`*`grid_control_hostname.domain`*`:`*`port`*`/em`

   If you are uncertain about the port number, you can refer to one of the following files:

   - *ORACLE_HOME*/install/setupinfo.txt, which includes information displayed by the Oracle Universal Installer at the end of the Grid Control install
   - *ORACLE_HOME*/install/portlist.ini on the Management Service host

2. When the Grid Control login page appears, enter the username and password for the Super administrator SYSMAN account, which you defined during the Grid Control installation.

   The Grid Control Console Home page appears.

3. From the **Target Search** section, select **Web Cache** from the **Search** list, and then click **Go** to display the OracleAS Web Cache targets.

   Enterprise Manager displays the the list of caches in the enterprise.

4. Select the cache you want to monitor.

   The Web Cache Home page (Figure 6–4 on page 6-7) appears.

Table 6–2 describes the main OracleAS Web Cache pages in the Grid Control Console.

Table 6–2   Grid Control Console Tabbed Pages for Managing Oracle Application Server

Enterprise Manager Web Cache Page	Description
Web Cache Home page	Use this page to gather overall status for the cache, such as its running status, availability, basic configuration settings, and hit rate.
	See Also: "Using the Grid Control Console: Web Cache Home Page" on page 6-6
Web Cache Performance page	Use this page to monitor the efficiency statistics of the cache.
	See Also: "Using the Grid Control Console: Web Cache Home Page" on page 6-6

Using the Grid Control Console: Web Cache Home Page

From the Web Cache Home page (Figure 6–4 on page 6-7), you can start and stop the cache, monitor the overall performance of the server, and monitor the hit rate of the cache.

Figure 6–4   Grid Control Console: Web Cache Home Page
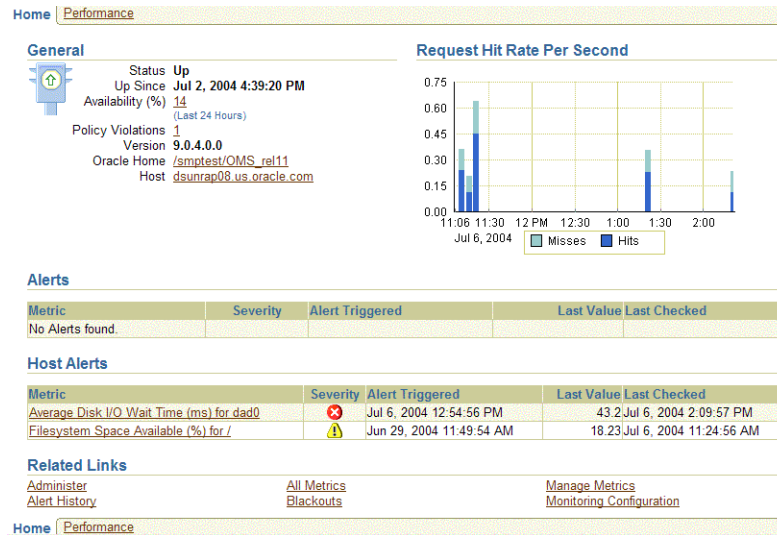
Using the Grid Control Console: Web Cache Performance Page

From the Web Cache Performance page (Figure 6–5 on page 6-8), you can view metrics related to the efficiency of the cache:

- CPU used by the cache

- Amount of data memory allocated for cache storage and operation

- Data served by the cache

- Client requests for data

- Percentage of requests which are served by cached content

You can also drill down to specific metrics dealing with data in the cache, including:

- Data stored in the cache

- Origin server usage by the cache

- Cached content served by the cache

- Error pages served by the cache

- Invalidation requests to cache and results

Figure 6–5   Grid Control Console: Web Cache Performance Page
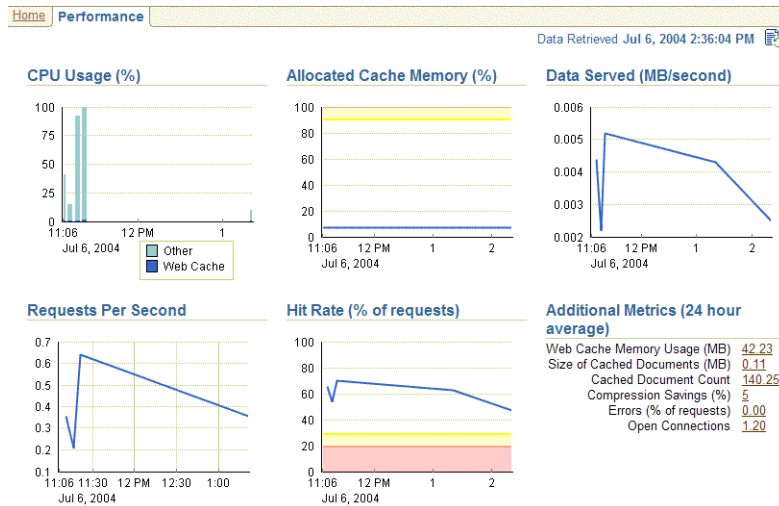
Using the Grid Control Console Online Help

You can click **Help** at the top of the page to get more information. In most cases, the Help window displays a help topic about the current page. Click **Help Contents** in the Help window to browse the list of help topics or to search for a particular word or phrase.

Managing Processes with Oracle Process Manager and Notification (OPMN)

Oracle Process Manager and Notification (OPMN) Server manages Oracle Application Server processes, including Oracle HTTP Server, OC4J, and OracleAS Web Cache processes, and channels all notifications from different components instances to all interested in receiving them.

With OPMN, you can administer the OracleAS Web Cache processes, including the **admin server process** and **cache server process**:

- The `admin` server process manages the OracleAS Web Cache Manager interface.

- The `cache` server process manages the cache.

The executable used for managing these process is `webcached`, which resides in `$ORACLE_HOME/webcache` directory on UNIX and the *ORACLE_HOME*`\bin` directory on Windows.

In addition, if auto-restart is enabled for the cache, OPMN monitors the `cache` server process. If the `cache` server process fails, OPMN restarts the process. The executable used for managing auto-restart is `webcachemon`, which resides in `$ORACLE_HOME/webcache` directory on UNIX and the *ORACLE_HOME*`\bin` directory on Windows.

> **See Also:** "Task 3: Configure Auto-Restart Settings" on page 8-10 for instructions on enabling auto-restart

To use OPMN, you use the `opmnctl` utility. First, you must start OPMN, using the following command:

```
opmnctl start
```

Then, you use OPMN to control OracleAS Web Cache. The following shows the format of the `opmnctl` commands:

```
opmnctl command [parameter=value] [parameter=value]
```

Table 6–3 shows the commands of the `opmnctl` utility that are applicable to OracleAS Web Cache.

Table 6–3   Commands of the opmnctl Utility

Command	Description
startproc	Starts the specified process or component.
stopproc	Stops the specified process or component. If used to stop the cache server process, this command also clears the cache of all content and all statistics. It waits for all currently accepted requests to be served, or until the user-specified timeout, before stopping the cache. To stop the specified process immediately, use the `WCShutdown=abort` parameter shown in Table 6–4.
restartproc	Stops, then restarts the specified process or component.
startall	Starts all processes controlled by OPMN.
stopall	Stops all processes controlled by OPMN.
status	Shows the status of the processes controlled by OPMN. For more information about the options for the status command, at the command line, enter: `opmnctl status -help`

Table 6–4 shows the parameters for the `opmnctl` utility. It also shows the valid values that are applicable for OracleAS Web Cache. Unless otherwise noted, you can use any parameter with any command, except for `status`, listed in Table 6–3.

Table 6–4   Parameters for the opmnctl Utility

Parameter	Valid Values	Description
ias-component=*value*	WebCache	Takes the specified action for the OracleAS Web Cache admin server process and cache server process. For example, the following command starts both the OracleAS Web Cache admin server and cache server processes: `opmnctl startproc ias-component=WebCache` You must always specify this parameter to administer any OracleAS Web Cache process.

process-type=*value*	WebCache
	
	WebCacheAdmin
		Takes the specified action for the process specified in the value:
		
		■ WebCache: The cache server process
		
		■ WebCacheAdmin: The admin server process
		
		The parameter ias-component=WebCache must precede this parameter. For example, the following command starts only the cache server process:
		
		`opmnctl startproc ias-component=WebCache`
		`process-type=WebCache`

WCShutdown=*value*	abort	Used only with the stopproc command. Aborts (immediately stops) the specified process or component. Note the following differences between a normal shutdown and an abort shutdown: During an normal shutdown, OracleAS Web Cache does not accept any new connections, but it satisfies the request for connections that were made before receiving the stopproc command. After the requests are satisfied, the cache shuts down. During an abort shutdown, OracleAS Web Cache does not accept any new connections. In addition, it drops all existing connections, even if the requests have not been satisfied. Then, the cache shuts down. The parameter ias-component=WebCache must precede this parameter.
WCCore=*value*	true	Enables OracleAS Web Cache to produce a core dump. The parameter ias-component=WebCache must precede this parameter. If OracleAS Web Cache generates a core file, contact Oracle Support.

The opmn executable is located in the $ORACLE_HOME/opmn/bin directory on UNIX and the ORACLE_HOME\bin directory on Windows.

> **Note:**   If you are running OracleAS Web Cache in a standalone environment (that is, you installed OracleAS Web Cache from a kit that included only this product; you did not install OracleAS Web Cache as part of an Oracle Application Server installation), you must use the webcachectl utility to administer OracleAS Web Cache processes.
>
> See Appendix B for information on using the webcachectl utility.

See Also:

- *Oracle Application Server Administrator's Guide* for additional information about OPMN

- "Task 2: Modify Security Settings" on page 8-7 to specify the user and group ID of the OracleAS Web Cache executables

- "Starting and Stopping OracleAS Web Cache In an Oracle Application Server Configuration" on page 7-1 to start or stop OracleAS Web Cache

Overview of Tools for Standalone Configurations

If you installed OracleAS Web Cache from a kit that included only this product or you did not install OracleAS Web Cache as part of an Oracle Application Server installation), then you must use the following standalone tools:

- Managing OracleAS Web Cache with OracleAS Web Cache Manager

- Managing Processes with webcachectl

Managing OracleAS Web Cache with OracleAS Web Cache Manager

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

- Start, stop, and restart OracleAS Web Cache

- Administer invalidation or rollover log files

- Monitor performance and requests

- Configure general properties of the cache

- Configure log settings

- Configure listening and operation port settings

- Configure origin server settings

- Configure site-specific settings

- Configure caching rules

- Associate caching rules with URLs

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

- Starting OracleAS Web Cache Manager

- Navigating OracleAS Web Cache

- Applying Static and Dynamic Configuration Changes

Starting OracleAS Web Cache Manager

To start OracleAS Web Cache Manager:

1. Start the **admin server process** with the following command:

```
opmnctl startproc ias-component=WebCache process-type=WebCacheAdmin
```

> **See Also:** "Managing Processes with Oracle Process Manager and Notification (OPMN)" on page 6-8 for information about the `opmnctl` command

2. Point your browser to the following URL:

   `http://web_cache_hostname:web_cache_port/webcacheadmin`

   By default, OracleAS Web Cache listens for administrative requests on port 9400. If this port is in use, the installation procedure attempts to assign other port numbers from a range from 9400 to 9499.

3. Enter either the Enterprise Manager user name, `ias_admin`, or the OracleAS Web Cache administrator user name, `administrator` and password.

   The password is the one you supplied during the installation.

   Note that changing the password for `ias_admin` through Enterprise Manager has no effect on the password for the OracleAS Web Cache.

   > **Note:** You can also point your browser to `http://web_cache_hostname:web_cache_port` to link to OracleAS Web Cache Manager, information about examples, user documentation, and the Oracle Technology Network.

   > **See Also:**
   >
   > - "Classes of Users and Their Privileges" on page 4-6 for further information about the administrator role
   >
   > - "Task 2: Modify Security Settings" on page 8-7 for information on modifying the administrator password
   >
   > - "Task 8: Configure OracleAS Web Cache with Operations Ports" on page 8-22 for information on modifying port 9400

Navigating OracleAS Web Cache

The OracleAS Web Cache Manager interface includes:

- Top frame containing **Apply Changes** and **Cancel Changes** buttons and OracleAS Web Cache status message

- Navigator frame with configuration and monitoring menu items

- Right frame with property sheet for selected menu item

Figure 6–5 on page 6-13 shows the OracleAS Web Cache Manager interface.

Figure 6–6   OracleAS Web Cache Manager Interface
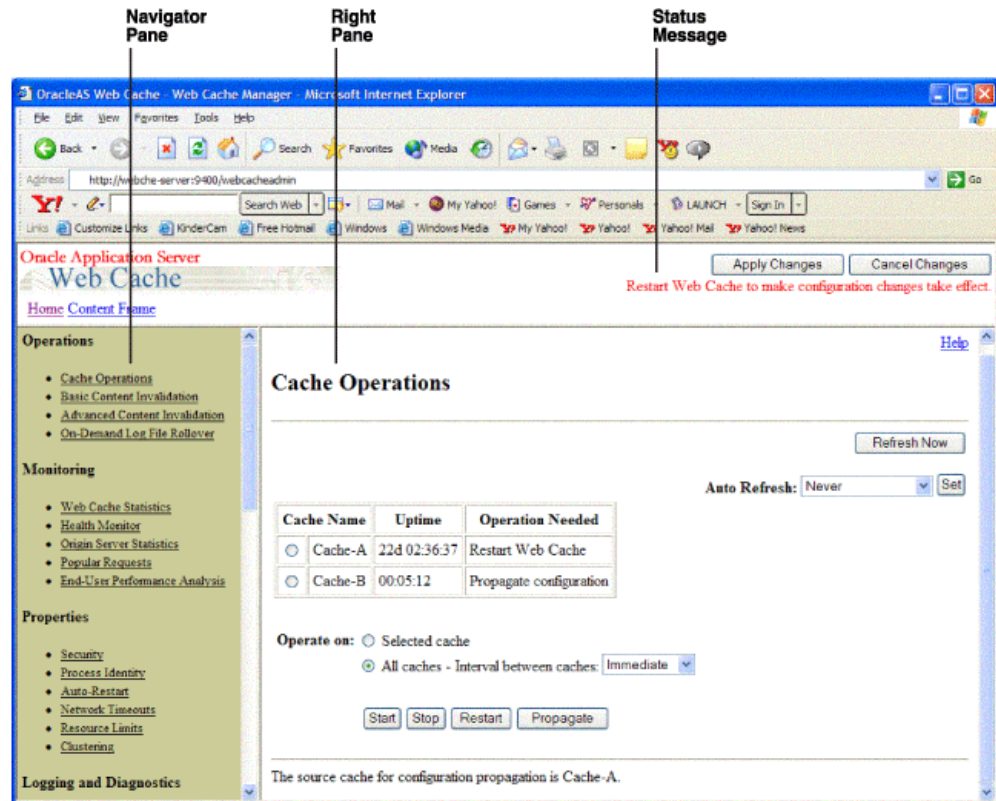
Apply Changes and Cancel Changes Buttons The **Apply Changes** button applies submitted static and dynamic configuration changes to OracleAS Web Cache; the **Cancel Changes** button cancels submitted static and dynamic configuration changes to OracleAS Web Cache.

> **See Also:**   "Applying Static and Dynamic Configuration Changes" on page 6-16 for further information about applying static and dynamic configuration changes

Status Messages Status messages appear beneath the **Apply Changes** and **Cancel Changes** buttons. Table 6–5 describes the possible status messages.

Table 6–5   OracleAS Web Cache Manager Status Messages

Message	Description
`Web Cache running with current configuration.`	This message appears if OracleAS Web Cache is running with an up-to-date configuration.
`Web Cache running in Load Balancer mode with current configuration.`	This message appears if OracleAS Web Cache is running as a software load balancer with an up-to-date configuration. **See Also:** "OracleAS Web Cache Solely as a Software Load Balancer or Reverse Proxy" on page 8-42 for information about configuring this mode
`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.

Table 6–5   (Cont.) OracleAS Web Cache Manager Status Messages

Message	Description
Restart Web Cache to make configuration changes take effect	This message appears if OracleAS Web Cache is running with an older version of the configuration. This can happen when static configuration changes have been applied to webcache.xml, but OracleAS Web Cache was not restarted.
Dynamic Changes Applied. Restart Not Needed	This message appears if one or more dynamic configuration changes were applied. **See Also:** "Applying Static and Dynamic Configuration Changes" on page 6-16
Retrieve configuration from remote cache	This message appears if the cache has been recently upgraded to the current version of OracleAS Web Cache but the configuration has not been copied to the local cache configuration file.

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

Navigator Frame

The navigator frame provides a graphical tree view of configuration, administration, and performance monitoring capabilities for OracleAS Web Cache and its supported Web sites.

The navigator frame contains the following major categories:

- Operations: This category contains pages that enable you to:
    - Start, stop, and restart OracleAS Web Cache
    - Invalidate objects in the cache
    - Rollover log files
- Monitoring: This category contains pages that enable you to:
    - Monitor the performance of OracleAS Web Cache
    - Monitor the performance of origin servers
    - View the contents of a cache
    - Analyze the performance of pages configured for end-user performance monitoring
- Properties: This category contains pages that enable you to:
    - Configure security settings such as usernames and passwords for administrators and invalidation users
    - Specify the process identity for OracleAS Web Cache processes
    - Specify settings for the auto-restart mechanism
    - Specify network time-outs
    - Specify the size of the cache and the number of connections
    - Configure a cache cluster
- Logging and Diagnostics: This category contains pages that enable you to:
    - Configure event logging settings
    - Configure access logging settings

- – Configure end-user performance monitoring

- – Configure diagnostic information

- Ports: This category contains pages that enable you to:

  - – Configure OracleAS Web Cache listening ports

  - – Configure listening ports for administration, statistics, and invalidation requests

- Origin Servers, Sites, and Load Balancing: This category contains pages that enable you to:

  - – Specify origin servers

  - – Configure site definitions

  - – Map site definitions to origin servers

  - – Configure error pages

  - – Configure session binding

  - – Specify the location of the wallet used for communication to an origin server

- Rules for Caching, Personalization, and Compression: This category contains pages that enable you to:

  - – Configure caching rules

  - – Specify expiration policies

  - – Configure session definitions

  - – Configure cookie definitions

- Rules Association: This category contains pages that enable you to associate caching rules with:

  - – Compression

  - – Expiration policies

  - – Session policies

  - – Cookies

  - – HTTP request headers

  - – HTTP errors

  - – ESI propagation policies

Right Frame

The right frame contains property sheets that enable you to configure and administer OracleAS Web Cache. Figure 6–7 on page 6-16 shows an excerpt of the Caching, Personalization, and Compression Rules page used for viewing caching rules.

Figure 6–7   Caching, Personalization, and Compression Rules Page

The Cache Operations Page

The Cache Operations page, shown in Figure 6–6 on page 6-13, of OracleAS Web Cache Manager (**Operations** > **Cache 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 Cache 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.

> **See Also:** "Starting and Stopping OracleAS Web Cache In an Oracle Application Server Configuration" on page 7-1 for further information about using the Cache Operations page

Applying Static and Dynamic Configuration Changes

Using the following menus in the navigator frame, you can change the configuration for OracleAS Web Cache:

- Properties
- Logging and Diagnostics
- Ports
- Origin Servers, Sites, and Load Balancing
- Rules for Caching, Personalization, and Compression
- Rule Association

Configuration changes are saved in a temporary configuration file, before they are applied to the `webcache.xml` file in the `$ORACLE_HOME/webcache` directory on UNIX or *ORACLE_HOME*`\webcache` directory on Windows.

Most configuration changes are static. When static changes are applied with **Apply Changes**, OracleAS Web Cache saves the temporary configuration file to the `webcache.xml` file. You must restart OracleAS Web Cache to apply changes.

However, OracleAS Web Cache recognizes some changes as dynamic. OracleAS Web Cache Manager provides dynamic configuration for the following features:

- Setting buffering and verbosity detail level in the Event Logs page (**Logging and Diagnostics** > **Event Logs**)

- Setting buffering in the Access Logs page (**Logging and Diagnostics** > **Access Logs**)

- Enabling and disabling of diagnostics information in the HTML response body of an object in the Diagnostics page (**Logging and Diagnostics** > **Diagnostics**)

- Setting routing to origin servers in the Origin Servers page (**Origin Servers, Sites, and Load Balancing** > **Origin Servers**)

OracleAS Web Cache Manager displays the icon, shown in Figure 6–8, to distinguish dynamic configuration from static configuration changes.

Figure 6–8   OracleAS Web Cache Manager Dynamic Configuration Icon

When dynamic changes are applied with **Apply Changes**, OracleAS Web Cache immediately applies the configuration changes and saves the temporary configuration file to the `webcache.xml` file. No restart is needed.

If you use OracleAS Web Cache Manager to apply both a dynamic and static configuration change, you must restart OracleAS Web Cache.

> **See Also:**   "Starting and Stopping OracleAS Web Cache In an Oracle Application Server Configuration" on page 7-1 for further information about restarting OracleAS Web Cache

Managing Processes with webcachectl

> **See Also:**   Appendix B for information about using the `webcachectl` utility

Understanding the OracleAS Web Cache Configuration Files

OracleAS Web Cache uses two configuration files: `webcache.xml` and `internal.xml`. Both Oracle Enterprise Manager 10*g* Application Server Control Console and OracleAS Web Cache Manager write configuration information to the `webcache.xml` file. OracleAS Web Cache uses `internal.xml` file for internal settings. 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 those special cases described in this guide, or when directed to do so by Oracle Support Services. Improper editing of these configuration files can cause result in errors.

Configuration and Administration Tasks at a Glance

OracleAS Web Cache configuration and administration tasks are described throughout this guide and in the OracleAS Web Cache Manager online help system. Table 6–6 lists the common tasks, and points you to the topic in this guide that describes the task.

Table 6–6   Common Administrative Tasks for OracleAS Web Cache

Task	See
Configuring OracleAS Web Cache Basics	
Change the `administrator` or `invalidator` password.	"Task 2: Modify Security Settings" on page 8-7
Configure the auto-restart mechanism.	"Task 3: Configure Auto-Restart Settings" on page 8-10
Modify the network time-outs for OracleAS Web Cache.	"Task 4: Configure Network Time Outs" on page 8-12
Set the maximum cache size limit.	"Task 5: Set Resource Limits" on page 8-13
Configure OracleAS Web Cache with listening ports.	"Task 6: Configure OracleAS Web Cache with Listening Ports for Client Requests" on page 8-19
	"Task 7: Provide Directives to Oracle HTTP Server" on page 8-22
	"Task 8: Configure OracleAS Web Cache with Operations Ports" on page 8-22
Specify the settings for origin servers.	"Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25
Configure site properties.	"Task 10: Configure Web Site Settings" on page 8-27
Configure error pages.	"Configure Error Pages" on page 8-37
Bind a session to an origin server.	"Bind a Session to an Origin Server" on page 8-39
Beyond Basics	
Configure support for HTTPS.	Chapter 9, "Configuring OracleAS Web Cache for HTTPS Requests"
Configure a cache cluster.	Chapter 10, "Configuring Cache Clusters"
Configure a cache hierarchy.	Chapter 11, "Configuring a Hierarchy of Caches"
Configure OracleAS Web Cache as a software load balancer.	"OracleAS Web Cache Solely as a Software Load Balancer or Reverse Proxy" on page 8-42
Configuring Logging and Diagnostics Settings	
Configure event log settings.	"Configuring Event Logs" on page 15-6
Configure access log settings.	"Configuring Access Logs" on page 15-22
Roll over log files.	"Rolling Over Event and Access Logs" on page 15-27
Configure end-user performance monitoring.	"Configuring End-User Performance Monitoring" on page 14-3
Administering OracleAS Web Cache	
Start and stop OracleAS Web Cache	"Starting and Stopping OracleAS Web Cache In an Oracle Application Server Configuration" on page 7-1
Invalidate objects in the cache.	Chapter 13, "Sending Invalidation Requests"
Propagate configuration changes to cache cluster members.	"Propagating Configuration Changes to Cache Cluster Members" on page 10-10
List the URLs of the objects in the cache.	"Listing Popular Requests and Cache Contents" on page 14-3

Table 6–6 (Cont.) Common Administrative Tasks for OracleAS Web Cache

Task	See
Monitoring Performance	
Monitor overall OracleAS Web Cache health.	"Monitoring OracleAS Web Cache Health" on page 14-2
Monitor OracleAS Web Cache performance.	"Viewing Cache Performance Statistics" on page 14-8
Monitor origin server performance.	"Viewing Origin Server Performance Statistics" on page 14-11

> **Note:** Except for those tasks listed under the **Administering OracleAS Web Cache** and **Monitoring Performance** rows, most tasks require stopping and then restarting OracleAS Web Cache. See "Applying Static and Dynamic Configuration Changes" on page 6-16 for exceptions, and "Starting and Stopping OracleAS Web Cache In an Oracle Application Server Configuration" on page 7-1 for further information about restarting OracleAS Web Cache.

Script for Setting File Permissions on UNIX

For UNIX operating systems, you can use the webcache_setuser.sh script to set the file permissions according to the mode in which in you need to run OracleAS Web Cache. The file webcache_setuser.sh is located in the directory $ORACLE_HOME/webcache/bin.

Prior to running the webcache_setuser.sh script, stop both the cache and admin server processes, using the OPMN utility command:

```
opmnctl stopproc ias-compononent=WebCache
```

You cannot use the options in the Cache Operations page to stop or start the admin server process. See "Starting and Stopping OracleAS Web Cache In an Oracle Application Server Configuration" on page 7-1 for more information on starting and stopping OracleAS Web Cache.

The following shows the format of the webcache_setuser.sh syntax:

```
webcache_setuser.sh command user_ID
```

Table 6–7 describes the commands.

Table 6–7 Commands of the webcache_setuser.sh Script

Command	Description
setroot	Sets the ownership of the webcached executable to root, and runs OracleAS Web Cache as the user that performed the installation.
setidentity	Changes the ownership of the runtime OracleAS Web Cache user. This command adds set-user ID permission to the webcached executable.
revert	Reverts the file permissions back to the installation state.
	It is necessary to revert permissions if you used the setidentity command and plan to install a patch release. Otherwise, you will be unable to write to files in the $ORACLE_HOME/webcache directory. After the patch installation is complete, you can choose to change the process identity again with the setidentity command.

The parameter *user_ID* is the user ID associated with the OracleAS Web Cache processes. (By default, that user ID is the ID of the user that performed the installation.) For setroot and revert modes, the user ID must be the ID of the user that performed the installation. The user ID must match the user ID specified in Security page (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Origin Servers**) of Application Server Control Console or the Process Identity page (**Properties** > **Process Identity**) of OracleAS Web Cache Manager.

> **See Also:** "Task 2: Modify Security Settings" on page 8-7 and "Running webcached with Root Privilege" on page 8-49 for further information about when running the webcache_setuser.sh script is necessary

7

Starting and Stopping OracleAS Web Cache

This chapter describes various procedures for starting and stopping OracleAS Web Cache.

This chapter contains these topics:

- Overview of Starting and Stopping for OracleAS Web Cache
- Starting and Stopping OracleAS Web Cache In an Oracle Application Server Configuration
- Starting and Stopping OracleAS Web Cache In a Standalone Configuration

Overview of Starting and Stopping for OracleAS Web Cache

Anytime the OracleAS Web Cache configuration is statically modified, you must stop and restart OracleAS Web Cache processes:

- The `admin` server process manages the administrative interface.
- The `cache` server process manages the cache.

The executable used for managing these processes is `webcached`, which resides in `$ORACLE_HOME/webcache` directory on UNIX and the `ORACLE_HOME\bin` directory on Windows.

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

Starting and Stopping OracleAS Web Cache In an Oracle Application Server Configuration

You can use the following tools to start, stop, restart, and view the status of components:

- `opmnctl`—a command-line tool
- Application Server Control Console—a Web-based tool

These tools are completely compatible—they both use OPMN as their underlying technology for managing processes—and can be used interchangeably. For example, you can start a component using `opmnctl` and stop it using the Application Server Control Console.

Although the two tools can be used interchangeably, they offer different features. The `opmnctl` command enables you to start and stop sub-processes within components, as well as the entire component. For example, you can start and stop OracleAS Web Cache, or you can start and stop only the `admin` server sub-process. With the

Application Server Control Console, you can view components that cannot be started or stopped, but whose status depends on other components. For example, the Application Server Control Console displays the status of the Single Sign-On component, whose status depends on the HTTP_Server component.

This section contains the following topics:

- Starting and Stopping Using opmnctl
- Starting and Stopping Using the Application Server Control Console
- Enabling and Disabling OracleAS Web Cache Using Application Server Control

Starting and Stopping Using opmnctl

To start, stop, or restart the OracleAS Web Cache processes with `opmnctl`:

1. Determine the status of OracleAS Web Cache. From the command line, enter:

   ```
   opmnctl status
   ```

   OPMN generates a list of processes that are running. The following message indicates that the OracleAS Web Cache `admin` server (`WebCacheAdmin`) and the `cache` server (`WebCache`) are already running:

   ```
   Processes in Instance: AppSrv.company.com
   ias-component      | process-type      |   pid | status
   -----------------------------------------------------------
   WebCache           | WebCacheAdmin     | 29121 | Alive
   WebCache           | WebCache          | 29120 | Alive
   OC4J               | OC4J_Demos        |   N/A | Down
   OC4J               | home              | 29268 | Init
   dcm-daemon         | dcm-daemon        | 29113 | Alive
   LogLoader          | logloader         |   N/A | Down
   HTTP_Server        | HTTP_Server       | 29099 | Alive
   ```

2. If the processes are not running, start the processes. From the command line, enter:

   ```
   opmnctl startproc ias-component=WebCache
   ```

   To stop the processes from the command line, enter:

   ```
   opmnctl stopproc ias-component=WebCache
   ```

   To restart the `admin` server and `cache` server processes from the command line, enter:

   ```
   opmnctl restartproc ias-component=WebCache
   ```

   > **See Also:**
   >
   > - "Managing Processes with Oracle Process Manager and Notification (OPMN)" on page 6-8 for a complete list of the `opmnctl` commands
   > - *Oracle Process Manager and Notification Server Administrator's Guide* to learn more about using `opmnctl`

Starting and Stopping Using the Application Server Control Console

To start, stop, restart, and view status of components on the Application Server home page:

1. Navigate to the Application Server home page on the Application Server Control Console. Scroll to the **System Components** section.

2. Select the checkboxes in the **Select** column for the components you want to start, stop, or restart.

3. Click the **Start**, **Stop**, or **Restart** button on the top right of the System Components section.

Enabling and Disabling OracleAS Web Cache Using Application Server Control

If you are not using OracleAS Web Cache in your configuration and want to conserve resources, consider disabling it. If you shut down OracleAS Web Cache without disabling it, the running status of the application server, even if all components are running, shows **Down** in the Application Server Home page. In the Application Server Home page, you can disable OracleAS Web Cache without removing its configuration settings. When you disable OracleAS Web Cache, it does not consume any system resources, and you can always enable it later.

Only disable OracleAS Web Cache for configurations in which it is not being used or is not needed. For example, in a configuration that uses non-cacheable OC4J applications that do not need compression, disable OracleAS Web Cache. Because OracleAS Portal requires OracleAS Web Cache, do not disable OracleAS Web Cache for OracleAS Portal configurations.

To disable OracleAS Web Cache:

1. From the **System Components** table in the Application Server Home page, click **Enable/Disable Components**.

   The Enable/Disable Components page appears.

2. From the **Enabled Components** list, select the Web Cache component and move it to the **Disabled Components** list.

3. Click **OK**.

4. If prompted to stop OracleAS Web Cache, click **Yes**.

   When the Application Server Home page refreshes, the **System Components** table no longer lists the Web Cache component. In addition, when you perform server-wide actions, such as **Start All**, **Restart All**, or **Stop All**, OracleAS Web Cache is not affected.

To re-enable OracleAS Web Cache:

1. From the **System Components** table in the Application Server Home page, click **Enable/Disable Components**.

   The Enable/Disable Components page appears.

2. From the **Disabled Components** list, select the Web Cache component and move it to the **Enabled Components** list.

3. Click **OK**.

4. When the Application Server Home page refreshes, the **System Components** table lists the Web Cache component.

> **See Also:** *Oracle Application Server Administrator's Guide* for further information about disabling and enabling components

Starting and Stopping OracleAS Web Cache In a Standalone Configuration

If you installed OracleAS Web Cache from a kit that included only this product or you did not install OracleAS Web Cache as part of an Oracle Application Server installation), you can use the following tools to start, stop, and restart:

- `webcachectl`—a command-line tool
- OracleAS Web Cache Manager—a Web-based tool

While `webcachectl` enables you manage all the OracleAS Web Cache processes, OracleAS Web Cache enables you to manage only the `cache` server process. To initialize OracleAS Web Cache for the first time, use the `webcachectl` utility rather than the OracleAS Web Cache Manager to start both the processes.

Starting and Stopping with webcachectl

> **See Also:** Appendix B for information about using the `webcachectl` utility

Starting and Stopping Using OracleAS Web Cache Manager

OracleAS Web Cache Manager enables you to start and stop the `cache` server process. You must use `webcachectl` to start, stop, or restart the admin server process in a standalone configuration.

To start, stop, or restart the `cache` server process with OracleAS Web Cache Manager:

1. Start OracleAS Web Cache Manager.

   See: "Starting OracleAS Web Cache Manager" on page 6-11

2. In the navigator frame, select **Operations** > **Cache Operations**.

   The Cache Operations page appears in the right pane.

3. In the Cache Operations page, select the cache and click **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 click **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 click **Start**, **Stop**, or **Restart**.

Part II

Configuration and Administration of OracleAS Web Cache

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

This part contains these chapters:

8

Setup and Configuration

This chapter describes the main tasks to begin caching content with OracleAS Web Cache.

This chapter contains these topics:

- Using the Default Configuration
- Tasks for Setting Up OracleAS Web Cache
- Configuring for High Availability Without a Hardware Load Balancer
- Advanced Configuration Topics

Using the Default Configuration

OracleAS Web Cache is installed with several default settings that you can either use or modify. Table 8–1 describes the main default configuration settings and where in the Oracle Enterprise Manager 10*g* Application Server Control Console and OracleAS Web Cache Manager interfaces you can change the values.

Table 8–1   OracleAS Web Cache Default Settings

Configuration Settings	Default Value	Location to Change Value in Oracle Enterprise Manager 10*g* Application Server Control Console and OracleAS Web Cache Manager
Security		
Password for the `administrator` account	Password entered for the administrator username during installation	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Security** OracleAS Web Cache Manager: **Properties** > **Security**
Password for the `invalidator` account	Password entered for the administrator username during installation.	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Security** OracleAS Web Cache Manager: **Properties** > **Security**

Table 8–1 (Cont.) OracleAS Web Cache Default Settings

Configuration Settings	Default Value	Location to Change Value in Oracle Enterprise Manager 10*g* Application Server Control Console and OracleAS Web Cache Manager
Process identity for OracleAS Web Cache (UNIX only)	User and group ID of user that installed OracleAS Web Cache	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Security** OracleAS Web Cache Manager: **Properties** > **Process Identity**
Auto-Restart	Enabled/Disabled: Enabled Failover threshold: 3 Ping URL: /_oracle_http_ server_webcache_static_ .html Polling interval for polling: 15 seconds Ping timeout: 30 seconds	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Auto-Restart** OracleAS Web Cache Manager: **Properties** > **Auto-Restart**
Network Timeouts		
Keep-Alive timeouts	5 seconds	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Resource Limits and Timeouts** OracleAS Web Cache Manager: **Properties** > **Network Timeouts**
Origin server timeout	3600 seconds	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Resource Limits and Timeouts** OracleAS Web Cache Manager: **Properties** > **Network Timeouts**
Resource Limits		
Maximum cache size	500 MB	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Resource Limits and Timeouts** OracleAS Web Cache Manager: **Properties** > **Resource Limits**
Maximum incoming connections	700	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Resource Limits and Timeouts** OracleAS Web Cache Manager: **Properties** > **Resource Limits**

Table 8–1   (Cont.) OracleAS Web Cache Default Settings

Configuration Settings	Default Value	Location to Change Value in Oracle Enterprise Manager 10*g* Application Server Control Console and OracleAS Web Cache Manager
Maximum size of single cached object	100 KB	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Resource Limits and Timeouts** OracleAS Web Cache Manager: **Properties** > **Resource Limits**
Logging and Diagnostics		
Event logs	`event_log` in `$ORACLE_HOME/webcache/logs` on UNIX and `ORACLE_HOME\webcache\logs` on Windows	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Logging** OracleAS Web Cache Manager: Logging and Diagnostics > **Event Logs**
Access logs	`access_log` in `$ORACLE_HOME/webcache/logs` on UNIX and `ORACLE_HOME\webcache\logs` on Windows	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Logging** OracleAS Web Cache Manager: **Logging and Diagnostics** > **Access Logs**
Ports		
OracleAS Web Cache	HTTP: 7777 on UNIX and 80 on Windows	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports** OracleAS Web Cache Manager: **Ports** > **Ports**
Administration	HTTP: 9400	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports** OracleAS Web Cache Manager: **Ports** > **Operations Ports**
Invalidation	HTTP: 9401	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports** OracleAS Web Cache Manager: **Ports** > **Operations Ports**
Statistics	HTTP: 9402	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports** OracleAS Web Cache Manager: **Ports** > **Operations Ports**
Origin Servers, Sites, and Load Balancing		

Table 8–1   (Cont.) OracleAS Web Cache Default Settings

Configuration Settings	Default Value	Location to Change Value in Oracle Enterprise Manager 10*g* Application Server Control Console and OracleAS Web Cache Manager
Oracle HTTP Server listening ports	HTTP: 7778 (for the first installation)	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Origin Servers** OracleAS Web Cache Manager: **Origin Servers, Sites, and Load Balancing** > **Origin Servers**
Load balancing and failover settings	Capacity: 100 Failover threshold: 5 Ping URL: / Polling interval for polling: 10 seconds	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Origin Servers** OracleAS Web Cache Manager: **Origin Servers, Sites, and Load Balancing** > **Origin Servers**
Site definitions	A default site definition is established for the Oracle HTTP Server when Oracle Application Server is installed ■ Host Name: `Oracle_HTTP_Server_host` ■ HTTP port: 7778 (for the first installation)	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites** OracleAS Web Cache Manager: **Origin Servers, Sites, and Load Balancing** > **Site Definitions**
Site-to-server mappings	■ Site host name and port `Oracle_HTTP_Server_host`:`Oracle_HTTP_Server_port` ■ Origin server host name and port `Oracle_HTTP_Server_host`:`Oracle_HTTP_Server_port`	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites** OracleAS Web Cache Manager: **Origin Servers, Sites, and Load Balancing** > **Site-to-Server Mappings**

Table 8–1   (Cont.) OracleAS Web Cache Default Settings

Configuration Settings	Default Value	Location to Change Value in Oracle Enterprise Manager 10*g* Application Server Control Console and OracleAS Web Cache Manager
Error Pages	■ Network error `network_error.html` in `$ORACLE_HOME/webcache/files` on UNIX and *ORACLE_HOME*`\webcache\files` on Windows ■ Site busy error `busy_error.html` in `$ORACLE_HOME/webcache/files` on UNIX and *ORACLE_HOME*`\webcache\files` on Windows ■ **Edge Side Includes (ESI)** default fragment `esi_fragment_error.txt` in `$ORACLE_HOME/webcache/files` on UNIX and *ORACLE_HOME*`\webcache\files` on Windows	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites** > **Default Error Pages** OracleAS Web Cache Manager: **Origin Servers, Sites, and Load Balancing** > **Error Pages**
Rules for Caching, Personalization, and Compression		

Table 8–1   (Cont.) OracleAS Web Cache Default Settings

Configuration Settings	Default Value	Location to Change Value in Oracle Enterprise Manager 10*g* Application Server Control Console and OracleAS Web Cache Manager
Caching rules	**See:** "Default Caching Rules" on page 12-5	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Rules** OracleAS Web Cache Manager: **Rules for Caching, Personalization, and Compression** > **Caching, Personalization, and Compression**
Expiration policies	■ As per HTTP Expires Header ■ After 300 seconds in cache ■ After 3600 seconds in cache	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Rules** > **Expiration Policies** OracleAS Web Cache Manager: **Rules for Caching, Personalization, and Compression** > **Expiration Policy Definitions**
Session definitions	Predefined site-specific session identifiers commonly used by components of Oracle Application Server: ■ 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 OracleAS Wireless session ID, PAconnxn is used for the OracleAS Wireless connection ID, and PAuserid is used for the OracleAS 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 Oracle Application Server Foundation Classes for persistent session tracking. The cookie name is ESFSID. There is no embedded URL parameter.	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sessions** OracleAS Web Cache Manager: **Rules for Caching, Personalization, and Compression** > **Session Definitions**

Tasks for Setting Up OracleAS Web Cache

To set up OracleAS Web Cache, perform the following tasks:

■ Task 1: Start OracleAS Web Cache

Task 1: Start OracleAS Web Cache

To start OracleAS Web Cache to begin the initial configuration:

1. If not currently logged on to the OracleAS Web Cache computer, log in with the user ID of the user that performed the installation.

2. Start both the `admin` and `cache` server processes:

   - The `admin` server process manages the administrative interface.

   - The `cache` server process manages the cache.

   Use `opmnctl` or `webcachectl` (for standalone installations) to start the processes:

   ```
   opmnctl startproc ias-component=WebCache
   ```

   ```
   webcachectl start
   ```

   > **See Also:** Chapter 7 for further information about starting and stopping the OracleAS Web Cache processes

Task 2: Modify Security Settings

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

To change the security settings in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Security**.

> **See Also:** "Modifying General Security Settings" in Enterprise Manager Online Help for instructions

To change the security settings in OracleAS Web Cache Manager:

1. Change the password for the OracleAS Web Cache usernames, `administrator` and `invalidator`.

   You can perform configuration and operational tasks with either the Application Server Control Console administrator, `ias_admin`, or the OracleAS Web Cache

administrator, `administrator`, accounts. Before you begin configuration, change the default `administrator` password to a secure password.

You can send invalidation requests with the `invalidator` username.

By default, the passwords for these users is the password you supplied during installation for the administrator username. In a **cache cluster**, the password for all **cluster members** must be the same.

a. In the navigator frame, select **Properties** > **Security**.

The Security page appears.

b. Under **Administration User**, click **Change Administration Password** or **Change Invalidation Password** under the **Invalidation User**.

c. When prompted, enter the old password in the **Old Password** field and a new password, in the **New Password** and **Confirm New Password** fields.

d. Click **Submit**.

> **See Also:** "Classes of Users and Their Privileges" on page 4-6 for further information about the administrator role

The passwords for the `administrator` and `invalidator` users have these restrictions:

- Passwords must be between 5 and 30 characters.

- At least one of the characters must be a number.

- Passwords can contain only alphanumeric and underscore (_) characters.

- Passwords must begin with an alphabetic character. Passwords cannot begin with a number, the underscore (_), the dollar sign ($), or the number sign (#).

- Passwords cannot be Oracle reserved words. The *Oracle Database SQL Reference* lists the reserved words. You can find this guide on Oracle Technology Network:
  http://www.oracle.com/technology/documentation/index.html

> **See Also:**
>
> - "Classes of Users and Their Privileges" on page 4-6 for further information about the different administrative roles
>
> - Section 7.3.3, "Portal Web Cache Settings" in *Oracle Application Server Portal Configuration Guide* for information about additional settings required for OracleAS Portal configurations

2. 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 OracleAS Web Cache is the trusted host.

a. In the Security page, click **Change Trusted Subnets** under the **Current Trusted Subnets.**

The Change Trusted Subnets dialog box appears.

b. Select one of the following options:

All subnets: Allows requests from all computers in all the subnets in the network.

This machine only: Allows requests from only this computer.

Enter list of IP addresses: Allows requests from all IP addresses you enter in a comma-delimited 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` sub-net 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 net-mask 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, the requests will fail.

---

   c. Click **Submit.**

3. Optionally, change the user ID and group ID for the OracleAS Web Cache processes on UNIX.

By default, the user that performed the installation is the owner of OracleAS Web Cache processes. This user can execute `opmnctl` commands. Users that belong to the same group ID of the user that performed installation can also execute `opmnctl` commands.

If the current `opmnctl` user does not match the configured user in the Process Identity page, the OracleAS Web Cache `webcached` executable must run as root. If the `webcached` executable is not able to run as root, error events are reported to the event log file, and OracleAS Web Cache fails to start.

> **See Also:** "Running webcached with Root Privilege" on page 8-49 for instructions on changing the `webcached` executable to run as root

To change the user ID and group ID for the OracleAS Web Cache processes on UNIX:

a. In the navigator frame, select **Properties** > **Process Identity**.

The Process Identity page appears.

b. Select the cache for which you want to modify settings, and then click **Change IDs.**

The Change Process Identity dialog box appears.

c. Enter the new user in the **User ID** field and the group ID of the user in the **Group ID** field.

d. Click **Submit.**

   e. Use the `webcache_setuser.sh` script as follows to change file and directory ownership:

   ```
   webcache_setuser.sh setidentity <user_ID>
   ```

   where `<user_ID>` is the user you specified in the **User ID** field of the Process Identity page.

   The `setidentity` command changes the ownership of the following files and directories to the new user ID:

   – `$ORACLE_HOME/webcache`

   – `$ORACLE_HOME/webcache/internal.xml`

   – `$ORACLE_HOME/webcache/webcache.xml`

   – `$ORACLE_HOME/webcache/webcache.xml.bak`

   – `$ORACLE_HOME/webcache/.webcache_tmp.xml.tmp`

   – `$ORACLE_HOME/webcache/logs/event_log_files`

   – `$ORACLE_HOME/webcache/logs/access_log_files`

   > **See Also:** "Script for Setting File Permissions on UNIX" on page 6-19 for further information about the `webcache_setuser.sh` script

If you change the `administrator` password or the trusted subnets in the Security page or any of the settings in the Process Identity page, then, after applying changes, restart both the **cache server process** and **admin server process** with the Oracle Process Manager and Notification (OPMN) Server utility command `opmnctl restartproc ias-component=WebCache` or the `webcachectl` utility command `webcachectl restart` (for standalone installations). You cannot use the **Restart** option in the Cache Operations page to restart the `admin` server process. Until the `admin` server process is restarted, you cannot submit invalidation requests from the Basic Content Invalidation or Advanced Content Invalidation pages.

> **See Also:** "Task 12: Restart OracleAS Web Cache" on page 8-42 for instructions on applying changes and restarting with command-line tools

Task 3: Configure Auto-Restart Settings

OracleAS Web Cache provides an auto-restart mechanism that checks that the **cache server process** is running and automatically restarts the `cache` server process if it is not running.

If auto-restart is enabled, the auto-restart mechanism restarts the `cache` server if it stops. By default, auto-restart is enabled.

> **Note:** If you installed OracleAS Web Cache as part of an Oracle Application Server installation, the auto-restart mechanism is run by Oracle Process Manager and Notification (OPMN) Server. If you installed OracleAS Web Cache from a kit that included only this product, the auto-restart mechanism is run by the OracleAS Web Cache monitor.

To change the auto-restart settings in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Auto-Restart**.

> **See Also:** "Modifying Auto-Restart Settings" in Enterprise Manager Online Help for instructions

To specify the settings for auto-restart in OracleAS Web Cache Manager:

1. In the navigator frame, select **Properties** > **Auto-Restart**.

   The Auto-Restart page appears.

2. To change the default settings, click **Edit.**

   The Edit Auto-Restart dialog box is displayed.

3. If auto-restart is not enabled, in the **Enabled** field, select **Yes.**

4. You can specify that the auto-restart mechanism polls (pings) the OracleAS Web Cache `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, the auto-restart mechanism restarts the cache server process.

   Note that the auto-restart mechanism will restart the `cache` server if it has failed, whether or not you enable pinging. However, if you enable pinging, the auto-restart mechanism will attempt to restart the cache if it receives network errors in response to its ping attempts.

   If you do not want the auto-start mechanism to actively ping the `cache` server, disable pinging by selecting **No** in the **Pinging Enabled** field.

   To enable pinging:

   a. In the **Pinging Enabled** field, select **Yes.**

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

      The default is three failures.

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

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

      Note: The threshold applies only to network errors and timeouts. If the `cache` server process is not running, the auto-restart mechanism immediately restarts the `cache` server.

      ---

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

      You *must* use a URL that is cacheable and is guaranteed to be stored in the cache. The default is `/_oracle_http_server_webcache_static_.html`, which is stored in the cache. If your environment uses a hardware load

balancer with pinging capability, then configure the load balancer with this ping URL to ping each OracleAS Web Cache server on a periodic basis.

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

The default value is 15 seconds.

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

The default value is 30 seconds.

5. Click **Submit.**

Task 4: Configure Network Time Outs

After OracleAS Web Cache sends a response to a client, the connection is left open for five seconds, which is typically enough time for the client to process the response from OracleAS Web Cache. If the network between the client and OracleAS Web Cache is slow, consider increasing the timeout. Likewise, there is a 3600 second network timeout between OracleAS Web Cache and the origin server. If the origin server is unable to generate a response within that time, OracleAS Web Cache drops the connection and sends a network error page to the client. If applications require a shorter timeout, adjust the timeout.

To modify the default network timeouts from the Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Resource Limits and Timeouts**.

> **See Also:** "Configuring Network Timeouts" in Enterprise Manager Online Help for instructions

To modify the default network timeouts from OracleAS Web Cache Manager:

1. In the navigator frame, select **Properties** > **Network Timeouts**.

The Network Timeouts page appears.

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

The Edit Network Timeouts dialog box appears.

3. In the **Keep-Alive Timeout** field, enter the time, in seconds, for OracleAS Web Cache to keep a connection open to the client after it has returned a response.

If the timeout is set to 0, the connection to the client is not kept open. In addition, OracleAS 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 OracleAS Web Cache. If the origin server is unable to generate a response within that time, OracleAS Web Cache drops the connection and sends a network error page to the client.

5. Click **Submit**.

> **See Also:**
>
> - "Task 6: Modify ssl.conf for Keep-Alive Connections" on page 9-6 for further information about configuring the Oracle HTTP Server to maintain keep-alive connections from OracleAS Web Cache for Internet Explorer browsers
>
> - "Browser Displaying a Page Not Displayed Error" on page E-10 for further information about the keep-alive limitations with Internet Explorer browsers

Task 5: Set Resource Limits

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

- Cache Memory
- Connection Limit
- Maximum Size of Single Cached Object

Cache Memory

When the maximum cache memory limit is reached, OracleAS Web Cache performs **garbage collection**. During garbage collection, OracleAS Web Cache removes stale objects based on **popularity** and **validity**. In a cache cluster environment, OracleAS Web Cache removes on-demand objects before it removes owned objects.

> **See Also:** "Performance Assurance Heuristics" on page 2-2 for further information on how OracleAS Web Cache determines when to serve stale content

To avoid swapping objects 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 OracleAS Web Cache should be set to at least 256 MB.

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

1. Determine which objects you want to cache, how many are smaller than 2 KB and how many are larger than 2 KB. Determine the average size of the objects that are larger than 2 KB. Determine the expected peak load—the maximum number of objects to be processed concurrently.

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

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

   The amount of memory that OracleAS Web Cache uses to store an object depends on whether the object is larger or smaller than 2 kilobytes (KB):

   - If an object is smaller than 2 KB, OracleAS Web Cache uses a buffer of 2 KB to store the HTTP body.

   - If an object is 2 KB or larger, OracleAS Web Cache uses buffers of 8 KB to store the HTTP body. For example, if an object is 42 KB, OracleAS Web Cache uses six 8 KB buffers to store the HTTP body.

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

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

```
(X * (2KB + 8KB)) + (Y * (([m/8] * 8KB) + 8KB)) + basemem
```

In the formula:

- *X* is the number of objects smaller than 2 KB.

- `2KB` is size of the buffer for the HTTP body for objects smaller than 2 KB.

- `8KB` is the size of the buffer for the HTTP response header.

- *Y* is number of objects that are 2 KB or larger.

- `[m/8]` is the ceiling of `m` (the average size, in kilobytes, of objects 2 KB or larger) divided by `8`. A **ceiling** is the closest integer that is greater than or equal to the number.

- `8KB` is size of the buffer for the HTTP body for objects that are 2 KB or larger.

- `8KB` is the size of the buffer for the HTTP response header.

- *basemem* is the base amount of memory needed by OracleAS 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 32 KB of memory. For example, to support 1000 concurrent requests, you need about 32 MB of memory.

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

  ```
  32KB + (number of ESI fragments * [8KB to 16KB])
  ```

  Because objects with more ESI fragments require more metadata for each fragment, use the higher number (16) for objects with 10 or more fragments. For example, for an object with 10 ESI fragments, use the following calculation:

  ```
  32KB + (10 * [16KB]) = 192KB
  ```

  That is, you need about 192 KB of memory for one 10-fragment object. To support 1000 concurrent requests, you need roughly 192 MB of memory.

For example, assume that you want to cache 5000 objects that are smaller than 2 KB and 2000 objects that are 2 KB or larger and that the larger objects have an average size of 54 KB. The objects do not use ESI. You expect to process 500 objects concurrently. Use the formula to compute the maximum memory:

```
(5000 * (2KB + 8KB)) + (2000 * (([54/8] * 8KB) + 8KB)) + (500 * 32KB)
```

Using the formula, you need:

- 50,000 KB for the smaller objects.

- 128,000 KB for the larger objects. For the HTTP body, you need 56 KB (seven 8 KB buffers) for each object, given the average size of 54 KB. For the HTTP response header, you need 8 KB for each object.

■ Approximately 16,000 KB for the base amount of memory needed to process 500 concurrent requests.

This results in an estimate of 194,000 KB of memory needed.

> **Note:** Even though you specify that certain objects should be cached, not all of the objects are cached at the same time. Only those objects that have been requested and are valid are stored in the cache. As a result, only a certain percentage of your objects 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 OracleAS Web Cache, specify the maximum cache size from the Resource Limits and Timeouts page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Resource Limits and Timeouts**) or the Resource Limits page of OracleAS Web Cache Manager (**Properties** > **Resource Limits**).

4. After applying changes and restarting OracleAS Web Cache, as described in "Task 12: Restart OracleAS Web Cache" on page 8-42, 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 OracleAS 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 objects are cached.

The Web Cache Statistics page of OracleAS Web Cache Manager (**Monitoring** > **Web Cache Statistics**) provides information about the current memory use and the maximum memory use. Note the following metrics in the Cache Overview table:

■ **Size of Objects in Cache** shows the current logical size of the cache. The logical size of the cache is the size of the valid objects in the cache. For example, if the cache contains two objects, one 3 KB and one 50 KB, the **Size of Objects 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 OracleAS Web Cache for cache storage and operation. This number is always smaller than the process size shown by operating system statistics because the OracleAS 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 95 percent 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**, OracleAS Web Cache begins garbage collection 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 objects in and out of the cache. Use the Resource Limits and Timeouts page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** >

Web Cache > **Resource Limits and Timeouts**) or the Resource Limits page of OracleAS Web Cache Manager (**Properties** > **Resource Limits**) to increase the maximum cache size.

Connection Limit

In addition to the cache size, it is important to specify a reasonable number for the maximum connection limit for the OracleAS 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 served. 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 from the Cluster Members and Properties page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Cluster Properties** > **Members and Properties**) or the Clustering page of OracleAS Web Cache Manager (**Properties** > **Clustering**).

Use various tools, such as those available with the operating system and with OracleAS 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. OracleAS Web Cache Manager provides statistics on hits and misses.

You set the maximum number of incoming connections from the Resource Limits and Timeouts page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Resource Limits and Timeouts**) or the Resource Limits page of OracleAS Web Cache Manager (**Properties** > **Resource Limits**).

Do not set the value to an arbitrarily high value, because OracleAS 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. OracleAS Web Cache tries to reserve the maximum number of file descriptors (`Max_File_Desc`) when it starts. If the OracleAS Web Cache `webcached` executable is run as root, you can increase this number. For example, on Sun Solaris, you can increase the maximum number of file descriptors by setting the `rlim_fd_max` parameter. If the `webcached` executable is not run with the root privilege, OracleAS Web Cache fails to start.

See Also:

- "Running webcached with Root Privilege" on page 8-49 for instructions on changing the `webcached` executable to run with the root privilege

- "Problem 5: More Than 1,024 Connections" on page E-4 for further information about the reported event log errors

OracleAS 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 OracleAS Web Cache. You set the maximum number of incoming connections from the Resource Limits and Timeouts page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Resource Limits and Timeouts**) or the Resource Limits page of OracleAS Web Cache Manager (**Properties** > **Resource Limits**).

  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 from the Cluster Members and Properties page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Cluster Properties** > **Members and Properties**) or the Clustering page of OracleAS Web Cache Manager (**Properties** > **Clustering**).

- `Total_WS_Capacity` is the sum of the capacity for all configured application Web servers. You set the capacity from the Origin Servers page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Origin Servers**) OracleAS Web Cache Manager (**Origin Servers, Sites, and Load Balancing** > **Origin Servers**).

  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)
```

You set the capacity of caches in a cluster from the Cluster Members and Properties page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Cluster Properties** > **Members and Properties**) or the Clustering page of OracleAS Web Cache Manager (**Properties** > **Clustering**).

- `100` is the number of connections reserved for internal use by OracleAS Web Cache.

> **See Also:**
> - Operating system-specific documentation for connection limitations
> - *Oracle 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 65536 ports.

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

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

Add a new value, specifying the following:

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

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 from the Resource Limits and Timeouts page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Resource Limits and Timeouts**) or the Resource Limits page of OracleAS Web Cache Manager (**Properties** > **Resource Limits**). You set the cluster member capacity from the Cluster Members and Properties page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Cluster Properties** > **Members and Properties**) or the Clustering page of OracleAS Web Cache Manager (**Properties** > **Clustering**).

On Windows operating systems, OracleAS 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.

Maximum Size of Single Cached Object

To conserve system resources, you can limit the size of objects that are cached, even if the objects meet other caching rules.

If you specify a maximum cached object size, only objects that are not larger than a specified size and that match the caching rules will be stored in the cache. Objects larger than the specified size will not be cached, even if they meet other caching rules. The default is 100 KB for 10*g* (9.0.4) and 10*g* Release 2 (10.1.2) installations. For upgraded caches, the default is that no limit is specified.

If you have objects that are larger than the maximum cached object size and those objects are requested frequently, consider increasing the limit. When you specify a value of 0, no objects will be cached.

If you want to apply the default to upgraded caches or change the limit, then modify the entry for the **Maximum Size of Single Cache Object** from the Resource Limits and Timeouts page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Resource Limits and Timeouts**) or the entry for **Maximum Cached Object Size** in the Resource Limits page of OracleAS Web Cache Manager (**Properties** > **Resource Limits**).

Task 6: Configure OracleAS Web Cache with Listening Ports for Client Requests

By default, OracleAS Web Cache listens with the HTTP protocol on port 7777 on UNIX and port 80 on Windows. If this port is in use, the installation procedure attempts to assign another port number from a range of 7777 to 7877.

You can add ports, if necessary. For example, it may be necessary to add an additional listening port if you want to assign OracleAS Web Cache a port that an origin server was previously listening on.

You can add a port and specify the HTTPS protocol to accept HTTPS client requests on that port.

> **See Also:** *Oracle Application Server Administrator's Guide* for information about updating the port numbers for other Oracle Application Server components

> **Note:** The IP address for the default HTTP port is set to * in Application Server Control Console and ANY in OracleAS Web Cache Manager. Upon startup, OracleAS Web Cache attempts to bind the port to all IP addresses. If multiple instances of OracleAS Web Cache are running on a multihomed host with multiple IP addresses, change * or ANY to a specific IP address to avoid port conflicts in the Ports page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**) or the Listen Ports page in OracleAS Web Cache Manager (**Ports** > **Listen Ports**).

This section contains the following topics:

- Adding a Listening Port
- Changing Site Configuration
- Configuring Other Oracle Application Server Components with the Listening Port

Adding a Listening Port

To specify a listening port from Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**.

> **See Also:** "Configuring Listen Ports" in Enterprise Manager Online
> Help for instructions

To specify a listening port from OracleAS Web Cache Manager:

1. In the navigator frame, select **Ports** > **Listen Ports**.

   The Listen Ports page appears.

2. In the Listen Ports page, click **Add**.

   The Edit/Add Listen Ports dialog box appears.

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

4. In the **IP Address** field, specify the computer running OracleAS Web Cache:

   - IP address written in a 32-bit dotted decimal notation

   - A host name that resolves to an IP address of the computer running OracleAS
     Web Cache. If you do not want to rely on Domain Name System (DNS) to
     resolve the host name, use a different name resolution mechanism, such as the
     UNIX `etc/hosts` file.

   - `ANY` to represent any IP address

5. In the **Port Number** field, enter the listening port from which OracleAS Web Cache
   will receive client requests for the Web site.

   Ensure that this port number is not already in use.

   Port numbers less than 1024 are reserved for use by privileged processes on UNIX.
   If you want to configure OracleAS Web Cache to listen on a port less than 1024,
   such as on port 80, run the OracleAS Web Cache `webcached` executable with the
   root privilege. If the `webcached` executable is not run as root, OracleAS Web
   Cache fails to start.

   > **See Also:**
   >
   > - "Running webcached with Root Privilege" on page 8-49 for
   >   instructions on changing the `webcached` executable to run as root
   >
   > - "Problem 4: Privileged Port Numbers" on page E-4 for further
   >   information about privileged ports and generated event log errors

6. From the **Protocol** list, select either **HTTP** to accept HTTP client requests on the
   port or **HTTPS** to accept HTTPS client requests on the port.

7. If you selected **HTTPS** as the listening protocol, you must configure additional
   information, including the location of the wallet:

   > **See:**
   >
   > - "Secure Sockets Layer (SSL)" on page 4-2 for further information
   >   about HTTPS
   >
   > - Chapter 9 for complete instructions on HTTPS configuration
   >   including creating wallets and specifying the wallet location

8. Click **Submit**.

Changing Site Configuration

In a configuration in which the OracleAS Web Cache listening port is the same as the logical site port, changing the OracleAS Web Cache listening port also requires you to change the logical site port in the Sites page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**) or the Site Definitions and Site-to-Server Mapping pages of OracleAS Web Cache Manager (**Origin Servers, Sites, and Load Balancing** > **Site-to-Server Mapping**).

When you change the OracleAS Web Cache listening port, you must also perform additional configuration for other Oracle Application Server components.

> **See Also:** "Creating Site Definitions" on page 8-28 for site configuration instructions

Configuring Other Oracle Application Server Components with the Listening Port

When you change the OracleAS Web Cache listen port, you must also perform additional configuration for other Oracle Application Server components.

> **See Also:** Section "Changing the OracleAS Web Cache Listen Port" in the *Oracle Application Server Administrator's Guide* for additional tasks required for other Oracle Application Server components

Task 7: Provide Directives to Oracle HTTP Server

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

- `Port=web_cache_port` specifies the OracleAS Web Cache listening ports, enabling dynamically created URLs to be redirected to OracleAS Web Cache.

- `Listen=Oracle_HTTP_Server_port` specifies the HTTP port 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_name
...
UseCanonicalName On
....
```

If you decide to disable OracleAS 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_name
...
UseCanonicalName On
....
```

If OracleAS Web Cache is deployed on a separate computer from the Oracle HTTP Server, then the Oracle HTTP Server administrator must create a `<VirtualHost>` container directive in `httpd.conf` for each site hosted by OracleAS Web Cache. This configuration enables Oracle HTTP Server to redirect URLs to OracleAS Web Cache. The following example shows `httpd.conf` modified to direct requests for `www.1st.company.com` and `www.2nd.company.com` to OracleAS Web Cache, which is listening on port 7777:

```
##
## httpd.conf -- Apache HTTP server configuration file
##
...
Port 7777
Listen 7778
...
ServerName  http_server_name
...
UseCanonicalName On
...
<VirtualHost *:*>
    ServerName www.1stcompany.com
    Port 80
    ...
</VirtualHost>
<VirtualHost *:*>
    ServerName www.2ndcompany.com
    Port 80
    ...
</VirtualHost>
...
```

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.

> **Note:** If you modify the value of `ServerName`, you must also modify the default site information for OracleAS Web Cache. See "Default Site Example" on page 8-28.

> **See Also:** *Oracle HTTP Server Administrator's Guide*

Task 8: Configure OracleAS Web Cache with Operations Ports

In addition to receiving HTTP and HTTPS client requests, OracleAS 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, OracleAS Web Cache listens with the HTTP protocol to receive these requests. The installation procedure assigns port numbers ranging from 9400 to 9499. Default HTTP port numbers are as follows:

- 9400 for administration requests

- 9401 for invalidation requests

- 9402 for statistics monitoring requests

> **Note:**
>
> - 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 (**Properties** > **Security**). See "Task 2: Modify Security Settings" on page 8-7 for further information.
>
> - By default, the administration, invalidation, or statistics monitoring ports are configured for HTTP basic authentication. The passwords for the `administrator` and the `invalidator` accounts 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.

This section contains the following topics:

- Modifying Operations Ports

- Configuring Other Oracle Application Server Components with Operations Ports

- Starting the admin Server Process After an Administration Port Change

- Starting the cache Server Process After an Invalidation or Statistics Port Change

Modifying Operations Ports

To modify the operations ports from Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**.

> **See Also:** "Configuring Operations Ports" in Enterprise Manager Online Help for instructions

To modify the operations ports from OracleAS Web Cache Manager:

1. In the navigator frame, select **Ports** > **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, click **Edit.**

   The Edit Operations Port dialog box appears.

4. Go to the **ADMINISTRATION**, **INVALIDATION**, or **STATISTICS** row.

5. In the **IP Address** field, specify the computer running OracleAS Web Cache:

   - IP address written in a 32-bit dotted decimal notation

- A host name that resolves to an IP address of the computer running OracleAS Web Cache. If you do not want to rely on Domain Name System (DNS) to resolve the host name, use a different name resolution mechanism, such as the UNIX `etc/hosts` file.

- `ANY` to represent any IP address

6. In the **Port Number** field, enter the operation port.

   Port numbers less than 1024 are reserved for use by privileged processes on UNIX. If you want to configure OracleAS Web Cache to listen on a port less than 1024, such as on port 80, run the OracleAS Web Cache `webcached` executable with the root privilege. If the `webcached` executable is not run as root, then error events are reported to the event log file, and the OracleAS Web Cache fails to start.

   > **See Also:**
   >
   > - "Running webcached with Root Privilege" on page 8-49 for instructions on changing the `webcached` executable to run as root
   >
   > - "Problem 4: Privileged Port Numbers" on page E-4 for further information about the reported event log errors

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

8. If you selected HTTPS as the listening protocol, you must configure additional information, including the location of the wallet.

   > **See Also:** Chapter 9 for complete instructions on HTTPS configuration

9. Click **Submit.**

Configuring Other Oracle Application Server Components with Operations Ports

When you change the operations ports, you must also perform additional configuration for other Oracle Application Server components.

> **See Also:** *Oracle Application Server Administrator's Guide* for additional tasks required for other Oracle Application Server components

Starting the admin Server Process After an Administration Port Change

After making an Administration port property change, restart the `admin` server process before you use the OracleAS Web Cache Manager for additional configuration. Use the following OPMN utility command:

```
opmnctl restartproc ias-component=WebCache process-type=WebCacheAdmin
```

If you are running a standalone instance of OracleAS Web Cache, use the following `webcachectl` utility command:

```
webcachectl restartadm
```

You cannot use the **Restart** option in the Cache Operations page (**Operations** > **Cache Operations**) to restart the `admin` server process.

Starting the cache Server Process After an Invalidation or Statistics Port Change

After making Invalidation or Statistics port property changes, restart the `cache` server process with the **Restart** option in the Cache Operations page. Until the `cache` server process is restarted after an Invalidation port change, you receive the following error when you submit invalidation requests from the Basic Content Invalidation or Advanced Content Invalidation pages (**Operations** > **Basic Content Invalidation** or **Advanced Content Invalidation**):

```
Internal error: can't connect to OracleAS Web Cache Invalidation Listening Port
```

Likewise, after a Statistics port change, the Cache Operations page does not display the current **Uptime** and **Operation Needed** information for the cache. In addition, the Monitoring pages (Web Cache Statistics, Health Monitor, Origin Server Statistics, and Popular Requests) report the following error.

```
Failure obtaining statistics from Web Cache cache server:  error in connect.
Please check that the cache server is up and running.
```

> **See Also:** "Task 12: Restart OracleAS Web Cache" on page 8-42 for instructions on applying changes and restarting the processes with command-line tools or OracleAS Web Cache Manager

Task 9: Configure Origin Server, Load Balancing, and Failover Settings

Configure OracleAS Web Cache with the application Web servers or proxy servers to which it sends cache misses. Typically, OracleAS Web Cache uses application Web servers for internal sites and proxy servers for external sites outside a firewall.

By default, the listening port and host name of the Oracle HTTP Server are configured. When OracleAS Web Cache is installed, Oracle HTTP Server has a default listening HTTP port of 7778 during the first installation sequence.

OracleAS Web Cache only forwards requests to a configured origin server if the server is mapped to a Web site. 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:**
>
> - "Stateless Load Balancing" on page 1-13 for an overview of load balancing
>
> - "Task 10: Configure Web Site Settings" on page 8-27 for instructions on configuring site-to-server mappings

To configure OracleAS Web Cache with origin server information from Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Origin Servers**.

> **See Also:** "Configuring Origin Servers" in Enterprise Manager Online Help for instructions

To configure OracleAS Web Cache with origin server information from OracleAS Web Cache Manager:

1. In the navigator frame, select **Origin Servers, Sites, and Load Balancing > Origin Servers**.

    The Origin Servers page appears.

2. Click **Add** in the **Application Web Servers** or **Proxy Servers** section.

The Add Application Web Server or 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 OracleAS Web Cache requests.

> **Note:** OracleAS Web Cache must listen on the same port as the application Web server being proxied. When configuring proxy servers, ensure there is a corresponding listening port for every port that will need to be proxied.

5. In the **Routing** field, select **ENABLED** to permit OracleAS Web Cache to route requests to the origin server or **DISABLED** to only serve requests from cache.

   Oracle recommends selecting **DISABLED** if temporary maintenance of an origin server is needed.

   OracleAS Web Cache tries to route a request matching a particular site to all origin servers mapped to that site. If all of the origin servers have a routing of **DISABLED**, OracleAS Web Cache serves a network error page to clients.

   > **See Also:** "Configure Error Pages" on page 8-37

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

   You determine this number 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, OracleAS 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
   ```

7. In the **Failover Threshold** field, enter the number of allowed continuous read/write failures with an origin server on established connections.

   The default is five request/response failures.

   If any connection failure occurs, OracleAS Web Cache immediately considers an origin server down.

   When the threshold is met, OracleAS Web Cache considers the origin server down and performs automatic **failover** of the origin servers. If an origin server fails at any time after OracleAS Web Cache has started to send a request, then OracleAS 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 other than connection failure errors.

   - The HTTP response status code is less than 100, or is one of the following messages: `500 Internal Server Error`, `502 Bad Gateway`, `503 Service Unavailable`, or `504 Gateway Timeout`.

   After the threshold is met, OracleAS Web Cache considers the server down and uses other servers for future requests. OracleAS Web Cache starts polling the down server, by sending requests to the URL specified in the **Ping URL** field. When OracleAS Web Cache receives a successful response from the server without

any network errors and the HTTP response code is not less than 100, or not equal to 500, 502, 503, 504, OracleAS Web Cache considers the server up again and uses it for future requests.

> **Note:**
>
> - The threshold does not apply if OracleAS Web Cache cannot connect to an origin server. In this case, OracleAS Web Cache immediately considers the server down and does not use it for future requests. If there are other origin servers, OracleAS Web Cache retries the request to another origin server. If there no servers configured, OracleAS Web Cache returns an error.
>
> - The failover to another origin server does not apply if there is only one origin server left.

8. In the **Ping URL** field, enter the URL that OracleAS 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 a fully-qualified URL that includes the domain name, or site name, representing the virtual host of the application Web server.

    - For a proxy server, 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 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. The default is "/".

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

    The default is 10 seconds.

10. 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)" on page 4-2

11. Click **Submit**.

12. If you selected HTTPS as the listening protocol, specify the location of the wallet for OracleAS Web Cache communication to the origin server (**Origin Servers, Sites, and Load Balancing** > **Origin Server Wallet**).

    > **See Also:** "Task 4: Configure HTTPS Port and Wallet Location for the Origin Server" on page 9-4 for information about configuring the wallet

Task 10: Configure Web Site Settings

For OracleAS Web Cache to act as a virtual server for one or more Web sites, configure OracleAS Web Cache with information about the Web site. To configure settings for a Web site, perform the following tasks:

- Creating Site Definitions
- Configure Error Pages

■ Bind a Session to an Origin Server

Creating Site Definitions

The installation process creates a default site definition that uses the host name and listening port of the computer on which Oracle HTTP Server was installed. OracleAS Web Cache forwards requests without host information to this site.

You need to create site definition for any named sites. A site definition consists of a host name, port information, and optional URL path prefix about the site and its aliases. Alias information is essential, because many sites are represented by one or more aliases. OracleAS 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, OracleAS 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, OracleAS Web Cache sends the request to the default site.

The following sections show examples of site definitions and explain how to configure new site definitions:

■ Default Site Example

■ Virtual Host Site Examples

■ ESI Provider Site Example

■ Creating New Site Definitions

Default Site Example For those requests that do not include a `Host` request-header field, OracleAS Web Cache sends the request to the default site. The default site, established during installation, uses the host name and listening port of the computer on which Oracle HTTP Server was installed. Figure 8–1 on page 8-29 shows an example of a default site definition from the Sites page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**).

The rules for the default site are as follows:

■ The site definition in the **Named Sites Definitions** section maps HTTP requests to the origin server `host-server:7778`. If you select to edit this definition, you can view the ESI content policy from the **Advanced** tab. The ESI content policy is set to **Unrestricted**, which instructs OracleAS Web Cache to serve site content, as well as assemble ESI include fragments.

■ The first mapping `*.80` in the **Server Mapping for Unnamed Site** uses a `*` wildcard host name to map all other virtual site names to the origin server. The ESI content policy is set to **Exclude Fragments**, which restricts OracleAS Web Cache from fetching ESI content from any sites other than the sites specified in the first rule.

■ The second mapping `*.*` in the **Server Mapping for Unnamed Site** uses a `*` wildcard host name to map all other virtual site names to the origin server and a `*` wildcard port number to map the site name to multiple port numbers. The ESI content policy is set to **Exclude Fragments**, which restricts OracleAS Web Cache from fetching ESI content from any sites other than the sites specified in the first two rules.

Figure 8–1   Default Site Definitions

Note: If you modify the name of the default site, you must also modify the `ServerName` directive in the `httpd.conf` file. See "Task 7: Provide Directives to Oracle HTTP Server" on page 8-21.

Virtual Host Site Examples   Figure 8–2 on page 8-30 shows how you would configure a virtual host site named `www.company.com:80` without ESI content from the Sites page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**).

The site definition in the **Named Sites Definitions** section maps HTTP requests to site `www.company.com:80` and and site alias `company.com:80` to origin server `host-server:7778`. The ESI content policy is set to **Exclude Fragments**, which restricts OracleAS Web Cache from fetching ESI content from `host-server:7778` for this site.

Figure 8–2   Example: Site Settings for a Virtual Host Site

Figure 8–3 on page 8-31 shows how you would configure a virtual host site named `www.1st.company.com:80` and `www.2nd.company.com` with ESI content from the Sites page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**).

The rules are as follows:

- The site definitions in the **Named Sites Definitions** section map HTTP requests to sites `www.1st.company.com:80` and `www.2nd.company.com:80` and respective aliases `1st.company.com` and `2nd.company.com` to origin server `host-server:7778`. The ESI content policy is set to **Unrestricted**, which enables OracleAS Web Cache to serve site content, as well as assemble ESI include fragments.

- The mapping `www.*.company.com:80` in the **Server Mapping for Unnamed Site** uses a `*` wildcard host name to map sites matching `www.*.company.com` to origin server `host-server:7778`. The ESI content policy is set to **Unrestricted**.

Figure 8–3   Example: Site Settings for Multiple Virtual Host Sites

ESI Provider Site Example Figure 8–4 on page 8-32 shows how you would configure ESI provider sites named `www.providersite1.com:80` and `www.providersite2.com` from the Sites page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**).

The site definitions in the **Named Sites Definitions** section specify how to map HTTP requests to sites `www.providersite1.com:80` and `www.providersite2.com:80` and respective aliases `providersite1.com` and `providersite2.com`. Requests to `www.providersite1.com` map to `proxy-host:80`. There is no origin server specified 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. The ESI content policy is set to **Fragments Only**, which restricts OracleAS Web Cache from using this mapping for any content that is not ESI.

Figure 8–4   Example: Site Settings for Multiple ESI Provider Sites

Creating New Site Definitions   If the default site definition is not adequate for your configuration, create additional site definitions.

To create new site definitions from Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**.

> **See Also:**   "Configuring Site Properties for a Named Site" or "Configuring Site Properties for an Unnamed Site" in Enterprise Manager Online Help for instructions

Using OracleAS Web Cache Manager, you can either discover site and alias information from Oracle HTTP Server, or you can manually create definitions. Oracle recommends using the site discovery feature to initially create the necessary definitions. After discovery is complete, you can manually create additional site definitions. Application Server Control Console does not support site discovery.

> **See Also:**   Help for the Site Definitions page in OracleAS Web Cache Manager Online Help for instructions on using the site discovery feature

To create new site definitions from OracleAS Web Cache Manager:

1. In the navigator frame, select **Origin Servers, Sites, and Load Balancing** > **Site Definitions**.

   The Site Definitions page appears.

> **Note:**
>
> - 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 OracleAS 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 OracleAS 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.

2. Specify the site information:

   a. In the Site Definitions page, click **Add Site**.

      The Add Site dialog box appears.

   b. In the **Host Name** field, enter the site host name, for example, `www.company.com`. To enable OracleAS Web Cache to match requests to this site, do not add protocol information (`http://` or `https://`) to the host name.

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

   c. Optionally, in the **URL Path Prefix** field, enter the path prefix of the URLs to distinguish this site from another site that shares the same host name. Ensure the prefix starts with "`/`" to distinguish from another site that shares the same host name. Do not include the file name or embedded URL parameters in the prefix.

      The prefix is excluded in matching requests to the site.

      For example, the following URLs share the same site name, but belong to two entirely differently applications hosted on entirely different origin servers:

      ```
      http://www.company.com/portal/page?_pageid=33,4232&_dad=portal
      http://www.company.com/um/traffic_cop?mailid=inbox
      ```

      While the first URL shows an email user a front page after login, the second URL displays an inbox. If the site host name is defined as `www.company.com`, then you specify the prefixes as `/portal` and `/um` to distinguish the sites. Requests are matched for `www.company.com`, not `www.company.com/portal` and `www.company.com/um`.

   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 client requests.

In a configuration in which the OracleAS Web Cache listen port is the same as the logical site port, changing the OracleAS Web Cache listen port requires you to also change the logical site port in both the Site Definitions and Site-to-Server Mapping pages.

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

f. In the **URL Parameter to Ignore** field, specify site-specific embedded URL or `POST` body parameters to ignore.

By configuring OracleAS Web Cache to ignore the value of embedded URL or `POST` body parameters, you enable OracleAS Web Cache to serve one cached object to multiple sessions requesting the same page. OracleAS Web Cache caches the response to the first request and serves subsequent requests for the page from its cache.

If you prefer instead to set a global parameter to apply to all sites, click **Global URL Parameters to Ignore** from the main **Site Definitions** page.

See Also:

- Excluding the Value of Embedded URL or POST Body Parameters on page 2-10 for an overview and an example scenario

- Configuring OracleAS Web Cache to Exclude the Value of Embedded URL or POST Body Parameters on page 12-18 for instructions on setting global parameters

g. In the **Client-Side Certificate** field, select **Required** or **Not Required** to specify that OracleAS Web Cache require or not require client-side certificates from client browsers.

A client-side certificate is a method for verifying the identity of the client. It binds information about the client user to the user's public key and must be digitally signed by a trusted CA.

h. In the **Default Site** field, select **Yes** to specify the site 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**.

If the default site includes a URL path prefix, the prefix is excluded in matching requests to the site. For example, if you specify a default site of `www.company.com` and a URL path prefix of `/portal`, requests are matched for `www.company.com`, not `www.company.com/portal`.

See Also: "Default Site Example" on page 8-28 for information about how the default site is used

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

– 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.com-pany.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.

3.   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, OracleAS Web Cache sends the request to the default site.

a.   In the Site Definitions page, select a site, and then click **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 client requests.

4.   Click **Submit.**

To map sites to origin servers from Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**.

> **See Also:**   "Configuring Site Properties for a Named Site" in Enterprise Manager Online Help for instructions

To map sites to origin servers from OracleAS Web Cache Manager, take the following steps:

1.   In the navigator frame, select **Origin Servers, Sites, and Load Balancing > Site-to-Server Mapping.**

The Site-to-Server Mapping page appears.

2.   Click **Create** if no mappings exist. If mappings already exist, select a mapping, and then click **Insert Above** or **Insert Below.**

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

3.   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.

> **Note:** 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 outside 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 with different port numbers to the same origin servers. If the origin servers are proxy servers, ensure they were configured to listen on the same port as the application Web server being proxied, as described in "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25.
>
> This option does not enable you to create a site definition. You must create a site definition in the Site Definitions page.

4. In the **Select either application Web servers or proxy 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 Origin Servers page

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

   > **Note:** If you configured multiple origin servers in "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25 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, load balancing for the site will not occur in the intended manner.

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

   - **Exclude Fragments** restricts OracleAS Web Cache from using this mapping for ESI fragments. Select this option if the site is a virtual host site that does not provide ESI content.

   - **Fragments Only** restricts OracleAS Web Cache from using this mapping for any content that is not an ESI fragment. Select this option if the site is an ESI provider site.

   - **Unrestricted** does not enforce any OracleAS 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 Fragments** does not mean that OracleAS Web Cache is not allowed to assemble ESI content from other origin servers.

6. Click **Submit.**

7. After you create the mappings, order them. For each request, the first matching mapping is used.

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

- Because mappings that use the wildcard `*` encompass a broader scope, give these mappings a lower priority than other mappings.

- Because requests are resolved to the first matching mapping, give mappings that contain the optional URL path prefix a higher priority than those mappings without an URL path prefix.

For example, you should order the following mappings as follows:

```
http://www.company.com/portal/page?_pageid=33,4232&_dad=portal
http://www.company.com/um/traffic_cop?mailid=inbox
http://www.company.com
```

If you instead reorder the mappings as follows, the request for URLs `http://www.company.com/portal/page?_pageid=33,4232&_dad=portal` and `http://www.company.com/um/traffic_cop?mailid=inbox` will never be reached. Requests for these URLs will instead resolve to `http://www.company.com` because it is listed first:

```
http://www.company.com
http://www.company.com/portal/page?_pageid=33,4232&_dad=portal
http://www.company.com/um/traffic_cop?mailid=inbox
```

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 OracleAS Web Cache uses the protocol configured for the origin server in the Site-to-Server Mapping page. OracleAS Web Cache also reports the following warning message to the event log:

```
ESI include fragment 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:

```
[25/Jul/2005:19:30:48 +0000] [warning 11250] [ecid: -]  ESI include
fragment protocol does not match origin server protocol: Origin
Server Protocol=http URL=https://www.company.com/gifs/frag1.gif
```

Configure Error Pages

For configured sites, specify error pages to be served from OracleAS Web Cache for network communication errors, site busy errors, and ESI `<esi:include>` errors:

1. Create error pages and place them in the `$ORACLE_HOME/webcache/files` directory on UNIX and the `ORACLE_HOME\webcache\files` directory on Windows. The default settings are as follows:

- For network errors, the default setting is set to `network_error.html`. This error 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 errors, the default setting is set to `busy_error.html`. This page is served when origin server capacity is reached.

- For ESI default fragments, the default setting is set to `esi_fragment_ error.txt`. This page is served when OracleAS Web Cache is unable to fetch the `src` specified in an `<esi:include>` tag and the `alt` attribute, `onerror` attribute, or the `try |attempt |except` block are either not present or fail.

> **See Also:** "Exceptions and Errors" on page 16-9

For a production environment, modify the defaults or create entirely new error pages to be consistent with other error pages for the site.

2. Configure OracleAS Web Cache with the error pages.

You can modify the settings for error pages in one of two ways:

- Change the default settings, and apply it to all defined sites.

- Modify the error page settings for a specific site.

From Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**.

> **See Also:** "Configuring Default Error Pages" in Enterprise Manager Online Help for instructions on configuring error pages

From OracleAS Web Cache Manager, perform these steps to configure error pages:

a. In the navigator frame, select **Origin Servers, Sites, and Load Balancing** > **Error Pages**.

The Error Pages page appears.

b. Select either *Default Pages* or a site name in the table, and then click **Edit**.

The Edit Error Pages dialog box appears.

c. In the **Network Error Page** field, enter the file name of the error page that will be delivered for network communication problems between OracleAS Web Cache and the Web site.

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

d. In the **Site Busy Page** field, enter the file name of the error page that will be delivered when a Web site is saturated with requests.

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

e. In the **ESI Default Fragment** field, enter the file name of the page that will be delivered when OracleAS 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 only ESI language elements for exceptions, do not enter a value.

f. Click **Submit**.

If you selected *Default Pages* in Step b, 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 b, the new settings will be applied to just to the site.

> **See Also:** "Exceptions and Errors" on page 16-9 to understand how exceptions and error are handled for `<esi:include>` errors

Bind a Session to an Origin Server

> **Notes:**
>
> - When a session cookie expires, OracleAS Web Cache does not continue to bind the user session to the origin server. Instead, OracleAS Web Cache uses load balancing to choose an origin server. To avoid pages being served past the client session expiration time, ensure that the session cookie expires before the origin server expires the client session.
>
> - If an origin server is busy, OracleAS Web Cache disables session binding to that origin server.

You can configure OracleAS 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.

If a request is forwarded to an origin server for an object requiring session binding, the origin server creates the user session by including the session information to clients through OracleAS Web Cache in the form of a **session cookie** or an embedded URL parameter. OracleAS Web Cache does not process the value of the parameter or cookie; it simply passes the information back to the client browser. When a client includes the session information in a subsequent request, OracleAS Web Cache forwards the request to the origin server that originally created the user session. OracleAS Web Cache binds the user session to that particular origin server.

If you have configured a cache cluster, you must enable tracking of session bindings through the use of a cookie that tracks session information so that it can be read by all cluster members. OracleAS Web Cache includes a `Set-Cookie` response-header in the response so that subsequent requests from the client include the cookie. The cookie provides information so that any of the cluster members can resolve the binding regardless of which cache handled the initial request.

> **See Also::**
>
> - "Session Binding (Stateful Load Balancing)" on page 1-16 for an overview of origin server binding
>
> - "Task 3: Enable Tracking of Session Binding" on page 10-7 for instructions on enabling session tracking in a cluster

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 Disable Session Binding. You can enable session binding in one of two ways:

- Change the default value of the *Default Session Binding* rule from Disable Session Binding to a specific session and binding mechanism. OracleAS Web Cache applies these settings to *all* sites that use the Default Session Binding rule. If you decide to change the value of the Default Session Binding rule, ensure all named sites currently configured with this rule require session binding. If some sites do

not require session binding, leave the value of Disable Session Binding, and instead specify session binding settings for the site. Use the next option.

■ Overwrite the *default session binding* setting to some other session for a specific site.

To enable session binding in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**.

> **See Also:** "Configuring Session Binding Settings" in Enterprise Manager Online Help for instructions

To enable session binding from OracleAS Web Cache Manager:

1. In the navigator frame, select **Origin Servers, Sites, and Load Balancing** > **Session Binding**.

   The Session Binding page appears.

2. In the Session Binding page, select *Default Session Binding* or a specific site name in the table, and then click **Edit Selected.**

   The Edit Session Binding dialog box appears.

3. From the **Please select a session** list:

   ■ If you selected the *Default Session Binding* rule in Step 2, change the session value from **Use Default Session Binding** to another defined session, and then skip to Step 6.

   ■ If you selected a specific site in Step 2, change the session value from Disable Session Binding to a defined session, and then skip to Step 5. Table 8–1 on page 8-1 provides descriptions of the default session definitions provided at installation time.

   If you want OracleAS Web Cache to bind user sessions with multiple cookies when any cookie is set, select a session of Any Set Cookie. When selecting Any Set Cookie, in **Please select a session binding mechanism**, select **Cookie-Based** to instruct OracleAS Web Cache to include a Set-Cookie response-header in the response.

   If the sessions listed do not contain the definition you require, click **Cancel** to exit the Edit Session Binding dialog box. Continue to Step 4.

4. Create a session definition:

   a. In the navigator frame, select **Rules for Caching, Personalization, and Compression** > **Session Definitions**.

      The Session Definitions page appears.

   b. From the **For Site** list, select the Web site for which you want to create site-specific site definitions.

   c. Click **Add** or **Create.**

      The Edit/Add Session Definition dialog box appears.

   d. In the **Session Name** field, enter an easy-to-remember unique name.

   e. Enter the cookie name in the **Cookie Name** field and the embedded URL parameter in the **URL or Post body 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:** OracleAS Web Cache requires a session cookie to perform session binding. If client browsers do not support cookies and you want to use an embedded URL parameter for the session, then perform the following for OracleAS 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.
>
> OracleAS 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://www.rfc-editor.org/` for further information about the `Set-Cookie` response header

---

    f. Click **Submit**.

    g. Repeat Steps 1 through 3.

5. From the **Please select a session binding mechanism** list, select one of the following mechanisms:

- **Cookie-based**: Select if the client supports cookies. OracleAS Web Cache uses its own cookie to track a session. This cookie, which contains routing information, is maintained between OracleAS Web Cache and the client browser.

- **OC4J-based**: Select if the application is based on OC4J. OracleAS Web Cache forwards routing information with each request to OC4J through Oracle HTTP Server.

- **Internal-tracking**: Select if the client does not support cookies and the application is not based on Oracle HTTP Server and OC4J. This option is intended for backward compatibility with earlier releases of OracleAS Web Cache. OracleAS Web Cache maintains an in-memory routing table, of which each entry maps a session ID to an origin server. The routing table is not shared among cluster nodes. If you select this option and you have a cache cluster configuration, then you must bind at the load balancer layer.

6. Click **Submit.**

If you selected the *Default Session Binding* rule in Step 2, 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 2, the new settings will be applied to just the site.

Task 11: Specify Caching Rules

Specify the URLs containing the objects you want OracleAS Web Cache to cache.

> **See:** "Configuring Caching Rules and Rule Association" on page 12-7

Task 12: Restart OracleAS Web Cache

After you configure OracleAS Web Cache, restart OracleAS Web Cache. To restart OracleAS Web Cache, use one of the following tools:

- For an Oracle Application Server installation, use either the Application Server Control Console or the command-line `opmnctl` utility.

  Both these tools enable you to restart the `cache` or `admin` server process, or both.

- For a standalone installation of OracleAS Web Cache, use either OracleAS Web Cache Manager or the command-line `webcachectl` utility.

  OracleAS Web Cache Manager enables you to restart only the `cache` server process; `webcachectl` enables you to restart the `cache` or `admin` server processes, or both.

You must restart *both* the `cache` server and `admin` server processes if you modified one of the following configuration settings:

- Administration port properties
- Password for the `administrator` account
- Trusted subnets
- User and group ID information

> **See Also:** Chapter 7 for an overview of starting, restarting, and stopping OracleAS Web Cache

Configuring for High Availability Without a Hardware Load Balancer

Many of the topologies described in Chapter 5 use a hardware **load balancer** to distribute incoming requests among origin servers. Instead, you can select to configure the following options:

- OracleAS Web Cache Solely as a Software Load Balancer or Reverse Proxy
- Operating System Load Balancing Support

OracleAS Web Cache Solely as a Software Load Balancer or Reverse Proxy

You can configure a special mode of OracleAS Web Cache that enables you to use OracleAS Web Cache solely as a software load balancer of HTTP traffic or **reverse proxy** to origin servers. This configuration mode is useful for:

- Managing traffic to low-volume, departmental, or test Web sites
- Proxying traffic in the DMZ between a hardware load balancer and an application using OracleAS Portal or OracleAS Single Sign-On

This mode disables all functions in OracleAS Web Cache, except for load balancing and SSL. For example, this mode disables support for the following features:

- Compression: OracleAS Web Cache ignores all compression settings.
- ESI: OracleAS Web Cache does not assemble ESI content.
- Cache hierarchies: If you plan to configure two caches in a cache hierarchy, then you should not configure one of the caches as a load balancer.

- Full cache cluster support: If you configure a cache cluster, OracleAS Web Cache automatically assigns a capacity of 0 to *all* cluster members, even if you configure other capacity values. When capacity is set to 0 for all cluster members, no requests will be forwarded between cluster members. However, you can still propagate the configuration and OracleAS Web Cache can automatically propagate invalidation requests to cluster members

- End-user performance monitoring: OracleAS Web Cache does not monitor the response time of your applications.

You can deploy a single OracleAS Web Cache server as a load balancer. However, this deployment makes the OracleAS Web Cache server a single point of failure for your application. You can instead configure a cache cluster with multiple OracleAS Web Cache servers in conjunction with operating system load balancing capabilities. Take note of the capacity changes mentioned earlier in this section.

In this mode, you can configure OracleAS Web Cache to load balance HTTP traffic in front of an application using ESI or in front of another OracleAS Web Cache. The OracleAS Web Cache load balancer does not process ESI content or participate in hierarchical caching. For example, a typical OracleAS Portal deployment has a built-in OracleAS Web Cache used for ESI assembly. For these configurations, do not configure the OracleAS Web Cache used for ESI assembly as a load balancer.

If you require other OracleAS Web Cache features, such as caching or compression support, do not configure this mode. Instead, configure a hardware load balancer or operating system load balancing support, and use the load balancing feature to manage requests to origin servers.

> **See Also:**
>
> - Operating System Load Balancing Support on page 8-45 for instructions on configuring operating system load balancing capabilities
>
> - "Stateless Load Balancing" on page 1-13 for configurations in which caching and load balancing support is needed
>
> - Chapter 10 for instructions on configuring a cache cluster
>
> - Chapter 11 for instructions on configuring a cache hierarchy
>
> - *Oracle Application Server Enterprise Deployment Guide* for instructions on using OracleAS Web Cache as reverse proxy for OracleAS Portal and OracleAS Single Sign-On

To configure a single OracleAS Web Cache server as a software load balancer:

1. Download an Automated Release Update (ARU) for bug 4569559.

   You can download ARUs from Oracle*Metalink*:

   http://metalink.oracle.com

2. Follow the instructions provided in the Readme for the patch.

3. Create a backup copy of the `internal.xml` file. This file is located in the `$ORACLE_HOME/webcache` directory on UNIX and `ORACLE_HOME\webcache` directory on Windows.

4. Use a text editor to open the `internal.xml` file.

5. Locate the `CALYPSOINTERNALPARAMS` element.

   . . .

```
    <CALYPSOINTERNALPARAMS>
      <HEURISTICS CATELMFACTOR="0.0"/>
      <CACHE/>
      <SEARCHKEY/>
      <INVALIDATION/>
      <MEMORYMANAGER/>
      <PPC/>
      <MISCELLANEOUS/>
      <OEMPERFTOOL/>
    </CALYPSOINTERNALPARAMS>
  ...
```

6. Add the LOADBALANCE subelement directly after the OEMPERFTOOL subelement as follows:

```
  ...
    <CALYPSOINTERNALPARAMS>
      <HEURISTICS CATELMFACTOR="0.0"/>
      <CACHE/>
      <SEARCHKEY/>
      <INVALIDATION/>
      <MEMORYMANAGER/>
      <PPC/>
      <MISCELLANEOUS/>
      <OEMPERFTOOL/>
      <LOADBALANCE ON="YES"/>
    </CALYPSOINTERNALPARAMS>
  ...
```

7. Save internal.xml.

8. Restart OracleAS Web Cache with the following command:

```
opmnctl restartproc ias-component=WebCache
```

9. Verify OracleAS Web Cache is running in the load balancer mode from the OracleAS Web Cache Manager by verifying the following status message displays beneath the **Apply Changes** and **Cancel Changes** buttons:

```
Web Cache running in Load Balancer Mode with current configuration
```

Application Server Control Console does not provide an equivalent verification status.

10. Ensure the auto-restart mechanism is enabled, as described in "Task 3: Configure Auto-Restart Settings" on page 8-10.

11. Configure origin servers, as described in "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25.

12. Create site definitions and map them to the origin servers, as described in "Task 10: Configure Web Site Settings" on page 8-27.

13. If your application deployment requires session stickiness, enable session binding. See "Bind a Session to an Origin Server" on page 8-39.

Applications using OracleAS Single Sign-On and Oracle Delegated Administration Services, for example, require session binding.

If the application that requires session stickiness is an OC4J application, then configure session binding using the JSESSIONID cookie.

Consider the topology depicted in Figure 8–5 on page 8-45.

Figure 8–5   Deploying OracleAS Web Cache as a Load Balancer
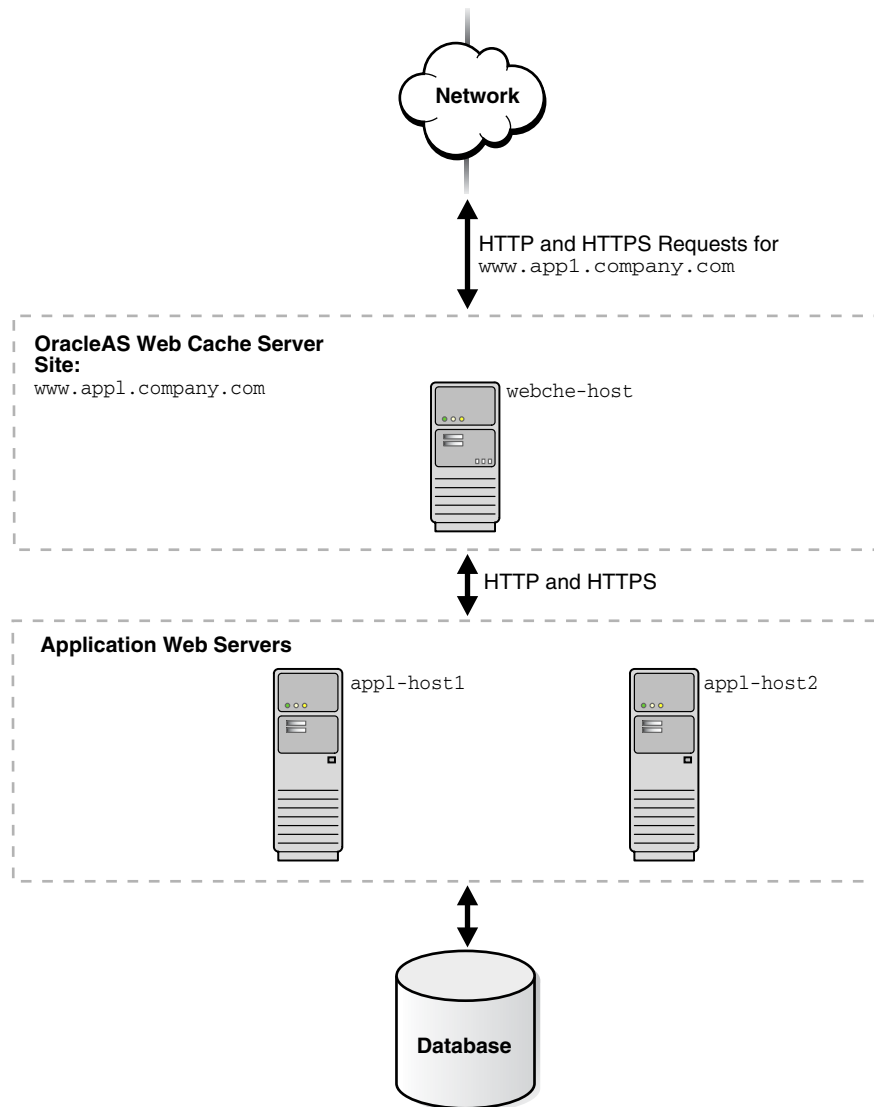
To configure this topology:

1. Register the IP address of the OracleAS Web Cache server `webche-host` with `www.app1.company.com`.

2. Configure the OracleAS Web Cache server with the following:

   a. Receive HTTP and HTTPS requests on designated listening ports.

   b. Send HTTP and HTTPS requests to application Web servers `app1-host1` and `app1-host2` on designated listening ports.

   c. Map virtual host site definition for `www.app1.company.com` mapped to `app1-host1` and `app1-host2`.

Operating System Load Balancing Support

Certain operating systems provide load balancing support, which can increase the availability of OracleAS Web Cache, particularly in cache clusters.

When the operating system detects a failure of one of the caches, automatic IP takeover is used to distribute the load to the remaining caches in the cluster configuration. Because requests are sent to the virtual IP address, not to a specific host, requests can be served even if one of the hosts is unreachable.

In addition, some operating systems provide load balancing for incoming requests. You can configure the operating system to balance the load of incoming requests across caches on multiple nodes.

A network load balancer does not provide all the features, such as firewall or ping URL mechanisms, that a hardware load balancer may provide, but if those needs are already met, you could consider using a network load balancer.

The following section, "Configuring Microsoft Network Load Balancing" describes how to configure a network load balancer on one operating system. Refer to your operating system documentation for more information.

Configuring Microsoft Network Load Balancing

On certain Windows platforms, you can use the Microsoft Network Load Balancing (NLB) component of the operating system instead of a hardware load balancer. NLB is part of the Microsoft clustering offerings and is available on the following platforms:

- Windows 2000 Advanced Server

- Windows 2000 Datacenter Server

- Windows 2003 (all editions)

You configure the hosts as a cluster and you configure the operating system to provide load balancing. Then, you configure NLB for hosts that are running Web Cache in a cache cluster, taking the following steps for each host:

1. Choose **Start** > **Settings** > **Network and Dial-up Connections**.

2. Select the network adapter. Then, right-click and select **Properties**.

3. In the Properties dialog box, select **Network Load Balancing**. Then, click **Properties**.

4. In the Cluster Parameters tab of the Network Load Balancing Properties dialog box, take the following steps:

   a. For **Primary IP Address**, enter the virtual IP address to be shared by all members of the cluster.

   b. For **Subnet mask**, enter the subnet mask for the virtual IP address.

   c. For **Full Internet Name**, enter the full internet name for the virtual IP address.

   d. Note the **Network Address**, which is a generated address.

   e. For **Multicast support**, check **enabled**.

   f. Optionally, enter a **Remote password** and enable **Remote control**.

5. Select the **Host Parameters** tab and take the following steps:

   a. For **Priority**, enter an integer between 1 and 32. The lower the number, the higher the priority. Priority establishes the default handling priority among hosts for requests that are not load-balanced by port rules. (See Step 6 for information about configuring port rules.)

   b. For **Initial cluster state**, check **active**. This specifies that this host should be included in the cluster array immediately upon Windows startup.

    c. For **Dedicated IP address**, enter the IP address of this host.

    d. For **Subnet mask**, enter the subnet mask of this host.

6. Select the **Port Rules** tab, and take the following steps:

    a. For **Port Range**, to balance the load from all client requests with a single port rule, use the default port range (1-65535). Use multiple port rules if different applications require different protocols, filtering modes, or affinity.

    b. For **Protocols,** select **TCP**. If your application uses software that requires UDP, select **Both.**

    c. For **Filtering Mode**, select **Multiple Hosts**.

    d. For **Affinity**, you can select one of three options. **None** results in load balancing of all requests across all hosts. **Single** results in all requests from a particular client being processed by the same host. Use this option to maintain session state. **Class C** results in all client requests from a TCP/IP class C address range being processed by the same host.

    e. For **Load Weight**, either enter a percentage of the load to be handled by the host or select **equal**.

Note that Port Rules must be identical for all hosts in the cluster.

For more information about Microsoft Network Load Balancing, see the Microsoft documentation at:

http://www.microsoft.com

Advanced Configuration Topics

This section contains these topics:

- Ensuring That ClientIP Headers Are Valid
- Configuring HTTP Request Header Limits
- Running webcached with Root Privilege

Ensuring That ClientIP Headers Are Valid

A client, such as a browser, can send information about its IP address in a header in a request. However, because a client could use a false IP address in the header, allowing a cache to forward that information to another cache or to the origin server can be a potential security problem. By default, OracleAS Web Cache removes any IP header information forwarded from a client and replaces it with a header that contains the correct IP address of the client. (In this case, a client can be a browser or another cache in a hierarchy.)

In a cache hierarchy, OracleAS Web Cache must be able to preserve the information that is forwarded from one cache to another in the hierarchy or from a cache to the origin server.

To configure these settings in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Security**.

> **See Also:** "Ensuring That ClientIP Headers Are Valid" in Enterprise Manager Online Help for instructions

To configure these settings in OracleAS Web Cache Manager:

1. In the navigator frame, select **Properties** > **Security**.

2. In the Special Security Header Configuration section of the Security page, check the value of the **Accept client IP addresses encoded in ClientIP headers** field.

   If the value is **NO**, OracleAS Web Cache removes any ClientIP request-header forwarded from the client and replaces it with a header that contains the correct IP address.

   If the value is **YES**, OracleAS Web Cache accepts the header received from the client and can forward it to another cache or the origin server.

3. If the settings do not match the following information, click **Edit** and change the settings in the Special Security Header Configuration dialog box:

   – For a simple configuration, the value should be **NO**.

   – In a cache cluster, the value should be **NO** for all cluster members.

   – In an ESI hierarchy, for the subscriber cache, the value should be **NO**.

   – In an ESI hierarchy, for the provider cache, the value should be **YES**.

     Because the provider cache received the header from the subscriber cache, it can safely forward it to the origin server.

   – In a distributed cache hierarchy, for the remote cache, the value should be **NO**.

   – In a distributed cache hierarchy, for a central cache that receives requests only from other caches, the value should be **YES**.

     If the central cache receives requests from both browsers and other caches in the hierarchy, OracleAS Web Cache cannot distinguish which is a browser and which is another cache. In this case, if you specify **YES**, a false IP address could potentially be forwarded from a browser. However, correct information would be forwarded from another cache. If you specify **NO**, a false IP address could not be forwarded from a browser. However, the information forwarded from another cache would contain the IP address of the cache, not of the original client.

4. Click **Submit**, and then click **Apply Changes**.

Configuring HTTP Request Header Limits

By default, OracleAS Web Cache provides the following limits for HTTP request header field:

- 819000 bytes for the total sum of all HTTP request header fields in requests

  Oracle recommends setting the header size to a lower value than the default to ensure security and prevent denial-of-service attacks from malicious clients.

  If the length of the request is larger than the allowed limit, OracleAS Web Cache sends an error to the client and reports the error 11356 to the event log:

  ```
  Total request header length exceeds configured maximum. A forbidden error
  response is returned to the client.
  ```

- 8152 bytes for an individual HTTP request header field

  Oracle recommends setting the individual header size based on how large an application sets HTTP requests header fields.

  If the length of the request is larger than the allowed limit, OracleAS Web Cache sends an error to the client and reports the error 11355 to the event log:

```
Single request header length exceeds configured maximum. A forbidden error
response is returned to the client.
```

To modify the default header limits in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Security**.

> **See Also:** "Configuring HTTP Request Header Limits" in Enterprise Manager Online Help for instructions

To modify the default header limits in OracleAS Web Cache Manager:

1. In the navigator frame, select **Properties** > **Security**.

   The Security page appears.

2. In the HTTP Request Header Limits section of the Security page, click **Edit**.

   The HTTP Request Header Limits dialog box appears.

3. In the **Maximum combined header size in bytes** field, specify the total sum of all HTTP request header fields in requests. Specify a limit of at least 4096 bytes (4 KB).

4. In the **Maximum individual header size in bytes** field, specify the allowed length limit of an individual HTTP request header fields. Specify a limit of at least 256 bytes.

5. Click **Submit**, and then click **Apply Changes**.

Running webcached with Root Privilege

On UNIX, you must configure `webcached` to run with root privilege in the following cases:

- Privileged port numbers less than 1024 are being used for OracleAS Web Cache listening ports.

- There are more than 1,024 file descriptors being used for connections to OracleAS Web Cache.

- The current `opmnctl` or `webcachectl` user does not match the configured process identity user in the Security page (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Origin Servers**) of Application Server Control Console or the Process Identity page (**Properties** > **Process Identity**) of OracleAS Web Cache Manager.

This section contains the following topics:

- Configuring Root Privilege for Privileged Ports and More than 1,024 File Descriptors

- Configuring Root Privilege for the Current User

- Reverting Permissions Back to Installation State

Configuring Root Privilege for Privileged Ports and More than 1,024 File Descriptors

For a configuration with privileged ports or to increase the file descriptor limit for OracleAS Web Cache, use the `setroot` command of `webcache_setuser.sh` to provide OracleAS Web Cache with root privilege without requiring changing the process identity settings:

1. From `$ORACLE_HOME/webcache/bin`, execute:

```
webcache_setuser.sh setroot user_ID
```

where *user_ID* is the user that performed installation.

> **See Also:** "Script for Setting File Permissions on UNIX" on page 6-19 for further information about the `webcache_setuser.sh` script

2. Log out of the computer, and re-login as the user that installed Oracle Application Server.

3. Start OracleAS Web Cache.

> **See Also:** Chapter 7, "Starting and Stopping OracleAS Web Cache"

Configuring Root Privilege for the Current User

For a configuration in which the current user does not match the configured user settings, change the process identity of the OracleAS Web Cache processes and use the `setidentity` command of `webcache_setuser.sh` to provide OracleAS Web Cache with root privilege:

1. Change the process identity of the OracleAS Web Cache processes.

   Oracle recommends running OracleAS Web Cache using a restricted user.

   > **See Also:** "Task 2: Modify Security Settings" on page 8-7 for instructions on setting the group ID and user ID to establish process identity

2. Use the `webcache_setuser.sh` script as follows to run OracleAS Web Cache as a different user and add set-user ID permission to the `webcached` executable:

```
webcache_setuser.sh setidentity user_ID
```

where *user_ID* is the user ID you specified in Step 2.

> **See Also:** "Script for Setting File Permissions on UNIX" on page 6-19 for further information about the `webcache_setuser.sh` script

3. Log out of the computer, and re-login as the user you configured in Step 2.

4. Start OracleAS Web Cache.

> **See Also:** Chapter 7, "Starting and Stopping OracleAS Web Cache"

Reverting Permissions Back to Installation State

You can revert permissions back to the installation state with the `revert` command of `webcache_setuser.sh`. It is necessary to revert permissions if you used the `setidentity` command and plan to install a patch release. Otherwise, you will be unable to write to files in the `$ORACLE_HOME/webcache` directory. After the patch installation is complete, you can choose to change the process identity again with the `setidentity` command.

To revert file permissions:

1. Use the `webcache_setuser.sh` script as follows to revert file permissions back to the installed state:

```
webcache_setuser.sh revert user_ID
```

where *user_ID* is the user that performed installation.

> **See Also:** "Script for Setting File Permissions on UNIX" on page 6-19
> for further information about the `webcache_setuser.sh` script

2. Log out of the computer, and re-login as the user that installed Oracle Application Server.

3. Start OracleAS Web Cache.

> **See Also:** Chapter 7, "Starting and Stopping OracleAS Web Cache"

9

Configuring OracleAS Web Cache for HTTPS Requests

To provide more security for your Web site, you can configure OracleAS Web Cache to receive **HTTPS protocol** client 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 OracleAS Web Cache and origin servers. You can also configure OracleAS Web Cache to send traffic to the origin server through an HTTPS listening port.

To configure HTTPS support, perform these tasks:

- Task 1: Create Wallets

- Task 2: Configure an HTTPS Listening Port

- Task 3: Configure HTTPS Operations Ports for the Cache

- Task 4: Configure HTTPS Port and Wallet Location for the Origin Server

- Task 5: Configure a Site to Accept HTTPS Requests

- Task 6: Modify ssl.conf for Keep-Alive Connections

- Task 7: (Optional) Require Client-Side Certificates

- Task 8: (Optional) Permit Only HTTPS Requests for a URL or Set of URLs

- Task 9: Restart OracleAS Web Cache

You can automate Tasks 2-6 and Task 9 by using the `SSLConfigTool` script.

> **See Also:** *Oracle Application Server Administrator's Guide* for information about the `SSLConfigTool`

Task 1: Create Wallets

To support HTTPS for OracleAS Web Cache, you must create a **wallet** on the OracleAS Web Cache server for each supported site. Wallets are needed to support the following HTTPS requests:

- Client requests for sites hosted by OracleAS Web Cache

- Administration, invalidation, and statistics monitoring requests to OracleAS Web Cache

- OracleAS Web Cache requests to origin servers, as well as `admin` server process requests for requests to invalidation and statistics monitoring ports enabled for SSL

A dummy wallet for the origin server is located in `$ORACLE_HOME/webcache/wallets/default` on UNIX and `ORACLE_HOME\webcache\wallets\default` on Windows. This wallet is intended for testing purposes. For a production environment, you must create a new wallet.

For each site that OracleAS Web Cache supports, configure at least one wallet. You specify the location of the wallet for each of the OracleAS Web Cache HTTPS listening and operations ports (to support incoming HTTPS requests), and the origin server (to support outgoing HTTPS requests). You can share one wallet, or you can create separate wallets. If you use the same wallet, keep in mind that it can support only one server-side certificate.

The following provides the basic steps for creating a wallet for use by OracleAS Web Cache. For detailed instructions, see the *Oracle Application Server Administrator's Guide*.

1. Invoke Oracle Wallet Manager:

   - On UNIX, run `owm` from `$ORACLE_HOME/bin`.

   - On Windows, choose **Start** > **Programs** > **Oracle - *Oracle_homename*** > **Network Administration** > **Wallet Manager**.

2. Create the wallet (**Wallet** > **New**), entering a password as prompted.

3. You are prompted whether or not to create a certificate request. Click **Yes.** Then, enter the information in the dialog box. For **Common Name,** specify the name or alias of the site that will be configured for HTTPS support.

4. Submit the certificate to a Certificate Authority (CA) for signature.

5. Import the CA's root certificate into the wallet (**Operations** > **Import Trusted Certificate**).

6. Enable Auto-login, which enables PKI-based access to services without a password. Select the wallet and choose **Wallet** from the menu bar. Check **Auto Login.**

7. Save the wallet. Select the wallet and choose **Wallet** > **Save**.

8. When you receive the signed certificate from the CA, import it into the wallet (**Operations** > **Import User Certificate**) and save the wallet.

By default, Oracle Wallet Manager stores wallets in the following locations:

- `/etc/ORACLE/WALLETS/user_name` on UNIX

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

> **See Also:**
>
> - "Secure Sockets Layer (SSL)" on page 4-2
>
> - "Problem 6: Opening Wallet" on page E-5

Task 2: Configure an HTTPS Listening Port

To configure HTTPS protocol support between client and OracleAS Web Cache, you must configure an HTTPS listening port for OracleAS Web Cache.

To configure an HTTPS listening port in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**.

> **See Also:** "Configuring Listen Ports" in Enterprise Manager Online Help for instructions

To configure an HTTPS listening port in OracleAS Web Cache Manager:

1. From the navigator frame in OracleAS Web Cache Manager, select **Ports** > **Listen Ports**.

2. Select a cache, and then click **Add**.

3. Specify the information for the port, selecting **HTTPS** for the **Protocol**. You must specify a port

4. Enable or disable client-side certificates. Select **Require Client-Side Certificate** to enable OracleAS Web Cache to require client browsers to provide SSL certificates.

    A client-side certificate is a method for verifying the identity of the client. It binds information about the client user to the user's public key and must be digitally signed by a trusted CA.

5. In the **Wallet** field, enter the directory location of the wallet. This directory must contain an existing wallet.

    This wallet is used for client requests for sites hosted by OracleAS Web Cache.

    You can share one wallet among all the HTTPS listening ports for a site and the origin server, or create separate wallets. If you use the same wallet, keep in mind that it can support only one server-side certificate.

6. Click **Submit**.

> **See Also:**
>
> - "Task 1: Create Wallets" on page 9-1 for further information about creating wallets and the option of sharing wallets among all listening ports, operations ports, and the origin server
>
> - "Task 6: Configure OracleAS Web Cache with Listening Ports for Client Requests" on page 8-19 for further information about adding listening ports
>
> - Section "Changing the OracleAS Web Cache Listener Port (Middle-Tier Installations)" in the *Oracle Application Server Administrator's Guide* for additional tasks required for other Oracle Application Server components

Task 3: Configure HTTPS Operations Ports for the Cache

To configure HTTPS ports to listen for administration, invalidation, or statistics monitoring requests in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**.

> **See Also:** "Configuring Operation Ports" in Enterprise Manager Online Help for instructions

To configure HTTPS ports to listen for administration, invalidation, and statistics monitoring requests in OracleAS Web Cache Manager:

1. From the navigator frame, select **Ports** > **Operations Ports**.

2. Select a cache, and then click **Edit Selected**.

3. Specify the information for the port, selecting **HTTPS** for the **Protocol.**

4. Enable or disable client-side certificates.

Select **Require Client-Side Certificate** to enable OracleAS Web Cache to require client browsers to provide SSL certificates.

A client-side certificate is a method for verifying the identity of the client. It binds information about the client user to the user's public key and must be digitally signed by a trusted CA.

5. In the **Wallet** field, enter the directory location of the wallet. This directory must contain an existing wallet.

This wallet is used for administration, invalidation, and statistics monitoring of HTTPS requests for sites hosted by OracleAS Web Cache.

You can share one wallet among all the HTTPS listening ports for a site and the origin server, or create separate wallets. If you use the same wallet, keep in mind that it can support only one server-side certificate.

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

6. Click **Submit.**

If you set an HTTPS invalidation or statistics monitoring port, you must configure a valid origin server wallet, as described in "Task 4: Configure HTTPS Port and Wallet Location for the Origin Server" on page 9-4. The `admin` server process requires this wallet to send HTTPS requests to invalidation and statistics monitoring ports enabled for SSL.

If you change the statistics protocol to HTTPS, it is not possible to view performance statistics in Enterprise Manager until a certificate is uploaded in Base64 format to `b64InternetCertificate.txt` to `$ORACLE_HOME/sysman/config` on UNIX and `ORACLE_HOME\sysman\config` on Windows.

> **See Also:**
>
> - "Task 1: Create Wallets" on page 9-1 for further information about creating wallets and the option of sharing wallets among all listening ports, operations ports, and the origin server
>
> - "Task 8: Configure OracleAS Web Cache with Operations Ports" on page 8-22 for further information about modifying operations ports
>
> - *Oracle Application Server Administrator's Guide* for additional tasks required for other Oracle Application Server components

Task 4: Configure HTTPS Port and Wallet Location for the Origin Server

You can configure HTTPS protocol support between OracleAS Web Cache and origin servers. When you use the Oracle HTTP Server as the origin server, requests from an OracleAS Web Cache server configured with an HTTPS listening port are passed on a secure (SSL) connection. It is not necessary to configure an HTTPS port for an Oracle HTTP Server. However, for other origin servers, you must configure an HTTPS port to secure the connection from OracleAS Web Cache to the origin server.

Then, you specify the location of the wallet for OracleAS Web Cache communication to the origin server. This wallet manages OracleAS Web Cache authentication data, such as keys, certificates, and trusted certificates needed by the **Secure Sockets Layer (SSL)**.

In addition to supporting OracleAS Web Cache requests using HTTPS to the origin server, this wallet also enables the `admin` server process to send HTTPS requests to invalidation and statistics monitoring ports enabled for SSL.

To configure HTTPS protocol support between OracleAS Web Cache and origin servers in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Origin Servers** to configure an origin server for HTTPS and **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Security** to configure the origin server wallet.

> **See Also:** "Configuring Origin Servers" and "Modifying General Security Settings" in Enterprise Manager Online Help for instructions

To configure HTTPS protocol support between OracleAS Web Cache and origin servers in OracleAS Web Cache Manager:

1. From the navigator frame, select **Origin Servers, Sites, and Load Balancing** > **Origin Servers**.

2. In the Origin Servers page, either click **Add** to add an origin server, or select an existing server and click **Edit.**

3. In the dialog box, specify the information for the origin server, selecting **HTTPS** for the **Protocol.** (See "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25 for information on configuring the origin server.)

4. Click **Submit.**

5. In the navigator frame, select **Origin Servers, Sites, and Load Balancing > Origin Server Wallet**.

   The Origin Server Wallet page appears.

6. Select the cache for which you want to modify wallet settings, and then click **Edit Selected.**

   The Edit Origin Server Wallet dialog box appears.

7. In the **Wallet Directory** field, enter the directory location of the wallet. This directory must contain an existing wallet.

   You can share one wallet among all the HTTPS listening ports for a site and the origin server, or create separate wallets. If you use the same wallet, keep in mind that it can support only one server-side certificate.

8. Click **Submit**.

   > **See Also:**
   > - "Task 1: Create Wallets" on page 9-1 for further information about creating wallets and the option of sharing wallets among all listening ports, operations ports, and the origin server
   > - "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25 for further information about modifying operations ports

Task 5: Configure a Site to Accept HTTPS Requests

You must specify a site that will accept HTTPS requests.

To configure site settings in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**.

> **See Also:** "Configuring Site Properties for a Named Site" in Enterprise Manager Online Help for instructions

To configure site settings in OracleAS Web Cache Manager:

1. In the navigator frame, select **Origin Servers, Sites, and Load Balancing** > **Site Definitions**.

2. In the Site Definitions page, click **Add Site**.

3. Specify the information, as described in "Task 10: Configure Web Site Settings" on page 8-27.

   In the **Port** field, enter the number of the HTTPS listening port. This site will use the wallet defined for that port.

   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.

4. Click **Submit.**

5. In the navigator frame, select **Origin Servers, Sites, and Load Balancing** > **Site-to-Server Mapping**.

6. Create a mapping from the site to an origin server, as described in "Task 10: Configure Web Site Settings" on page 8-27.

7. Click **Submit.**

Task 6: Modify ssl.conf for Keep-Alive Connections

By default, Oracle HTTP Server does not maintain keep-alive connection for HTTPS client requests from Microsoft Internet Explorer 5.5 and later releases. Internet Explorer has known issues with trying to reuse SSL connections after they have timed out. In order for Oracle HTTP Server to maintain keep-alive connections from OracleAS Web Cache, you must remove the following entry from the `ssl.conf` file in `$ORACLE_HOME/Apache/Apache/conf` directory on UNIX or `ORACLE_HOME\Apache\Apache\conf` directory on Windows:

```
SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown
```

The `ssl.conf` file specifies the SSL definitions for Oracle HTTP Server. If this entry is not removed, then keep-alive connections are disabled.

> **See Also:**
>
> - "Task 4: Configure Network Time Outs" on page 8-12 for further information about configuring the keep-alive timeout in OracleAS Web Cache
>
> - "Browser Displaying a Page Not Displayed Error" on page E-10 for further information about keep-alive limitations with Internet Explorer browsers

Task 7: (Optional) Require Client-Side Certificates

You can require that clients send certificates (client-side certificates) to the cache to verify the identity of the client.

With client-side certificates, the client browser sends the certificate to the cache during the SSL handshake. Then, the server processes the request for the object. If the requested object is not stored in the cache, the cache forwards the request to the application Web server, a peer cache (in a cluster), or a subordinate cache (in a hierarchy). To transfer information about the client-side certificate to another cache or

to the application Web server, OracleAS Web Cache adds HTTP headers to the request. The headers begin with the string `SSL-Client-Cert`.

Note the following points about using client-side certificates:

- In a simple configuration (client to cache to application Web server), the client sends the certificate to the cache during the SSL handshake. If the requested object is not stored in the cache, the cache forwards the request to the application Web server and transfers the client-side certificate information in headers to the application Web server. The application Web server recognizes the headers and responds to the request.

- In a cluster, the client sends the certificate to a cache cluster member during the SSL handshake. If the requested object is not stored in that cache, the cluster member requests it from a peer (the cluster member that owns the object). With client-side certificates, OracleAS Web Cache must be able to pass the client-side certificate information in headers to the peer cluster member, and the peer must be able to pass the headers to the application Web server.

- In an ESI hierarchical deployment, the client browser sends the certificate to the subscriber cache in a hierarchy. That cache must be able to forward the certificate information in headers to a provider cache. However, with this configuration, the provider caches could inadvertently accept the certificate information in a header from a bogus entity. To prevent this, you must secure the provider caches, by methods such as installing them behind a firewall.

- If client-side certificates are required, but not provided by the client, OracleAS Web Cache returns an error: `403: Forbidden.`

> **Note:** OracleAS Web Cache supports the use of client-side certificates with Oracle HTTP Server only.
>
> OracleAS Web Cache does not support client-side certificates with a distributed cache hierarchy because the security of the certificates cannot be guaranteed.

The following topics describe how to configure client-side certificate settings:

- Configuring Client-Side Certificate Settings for the HTTPS Listening Port
- Configuring Client-Side Certificate Settings for Cache Clusters
- Configuring Client-Side Certificate Settings for an ESI Cache Hierarchy
- Configuring Client-Side Certificate Settings for a Site

Configuring Client-Side Certificate Settings for the HTTPS Listening Port

To use client-side certificates, you must enable an HTTPS listening port, as described in "Task 2: Configure an HTTPS Listening Port" on page 9-2. If you have a cache cluster, you must enable HTTPS listening ports for all cluster members. In addition, you must configure OracleAS Web Cache to require client browsers to provide SSL certificates.

To enable this setting in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Security**.

> **See Also:** "Configuring Listen Ports" in Enterprise Manager Online Help for instructions

To enable this setting in OracleAS Web Cache Manager:

1. In the navigator frame, select **Ports** > **Listen Ports**.

   The Listen Ports page is displayed.

2. Select the HTTPS port and click **Edit.**

3. In the Edit Listening Port dialog box, select **Require Client-Side Certificate.**

4. Click **Submit.**

If you have a simple configuration, not a cache cluster or a cache hierarchy, proceed to the next section, "Task 8: (Optional) Permit Only HTTPS Requests for a URL or Set of URLs" on page 9-9.

After configuring the client-side certificate, to enable OracleAS Web Cache to transfer certificate information to Oracle HTTP Server, add the AddCertHeader directive to httpd.conf.

> **See Also:** *Oracle HTTP Server Administrator's Guide* for information about adding the AddCertHeader directive

Configuring Client-Side Certificate Settings for Cache Clusters

If you have a cache cluster, you must prevent a cache from accepting the certificate information in HTTP headers from any source other than a peer cluster member. In addition, each cache must be able to pass the client-side certificate information in headers to the peer cluster member, and the peer must be able to pass them to the application Web server.

To configure this behavior in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Security**.

> **See Also:** "Ensuring that ClientIP Headers Are Valid" in Enterprise Manager Online Help for instructions

To configure this behavior in OracleAS Web Cache Manager:

1. In the navigator frame, select **Properties** > **Security**.

2. In the Special Security Header Configuration section of the Security page, the value of the **Accept SSL client certificates encoded in SSL-Client-Cert HTTP headers** must be **NO** (the default):

   If it is not, click **Edit** to modify the setting in the Special Security Header Configuration dialog box.

3. In the Cluster Security Configuration, value of the **Route requests that contain SSL client certificates to cache cluster peers** must be **YES**.

   If it is not, click **Edit** to modify the setting in the Cluster Security Configuration dialog box.

Configuring Client-Side Certificate Settings for an ESI Cache Hierarchy

If you have an ESI cache hierarchy, a provider cache must be able to accept the client-side certificate information in headers from the subscriber cache.

To enable this behavior in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Security**.

> **See Also:** "Ensuring that ClientIP Headers Are Valid" in Enterprise Manager Online Help for instructions

To enable this behavior in OracleAS Web Cache Manager:

1. In the navigator frame, select **Properties** > **Security**.

2. In the **Special Security Header Configuration** section of the Security page, the value of Accept SSL client certificates encoded in `SSL-Client-Cert` HTTP headers must be **YES**.

   If it is not, click **Edit** to modify the setting in the Special Security Header Configuration dialog box.

3. If the subordinate caches are in a cluster, the subordinate caches must be able to pass the client-side certificate information in headers to the peer cluster member. In this case, in the Cluster Security Configuration section of the Security page, the value of the **Route requests that contain SSL client certificates to cache cluster peers** must be **YES**.

   If it is not, click **Edit** to modify the setting in the Cluster Security Configuration dialog box.

Configuring Client-Side Certificate Settings for a Site

You can also specify that an entire site require client-side certificates:

To configure a site to use client-side certificates in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**.

> **See Also:** "Configuring Site Properties for a Named Site" in Enterprise Manager Online Help for instructions

To configure a site to use client-side certificates in OracleAS Web Cache Manager:

1. In the navigator frame, select **Origin Servers, Sites, and Load Balancing** > **Site Definitions**.

2. In the Site Definitions page, select the site and click **Edit Site.**

3. In the Edit Site page, select **Required** in the **Client-Side Certificate** field.

4. Click **Submit.**

Task 8: (Optional) Permit Only HTTPS Requests for a URL or Set of URLs

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 8-27.

2. In Step 2e, enter the URL or URL prefix.

   If all traffic must be restricted to HTTPS, enter "/" for the entire site.

Task 9: Restart OracleAS Web Cache

If you are using OracleAS Web Cache Manager, click **Apply Changes** in the main window.

After you make configuration changes, you must restart the `cache` or `admin` server processes, using the `opmnctl` utility or `webcachectl` utility (for standalone

installations) on the computer on which OracleAS Web Cache software is installed and configured.

> **See Also:**   Chapter 7, "Starting and Stopping OracleAS Web Cache"

10

Configuring Cache Clusters

This chapter provides instructions for configuring a **cache cluster**, also known as OracleAS Cluster (Web Cache).

This chapter contains these topics:

- Configuring a Cache Cluster
- Removing Caches from a Cluster
- Configuring Administration and Invalidation-Only Clusters
- Propagating Configuration Changes to Cache Cluster Members

> **See Also:** Chapter 3 for an overview of cache clusters

Configuring a Cache Cluster

To increase the availability and scalability of your Web site, you can configure multiple instances of OracleAS Web Cache to run as members of a cache cluster.

To configure a cache cluster, you configure two or more OracleAS Web Cache instances as cache cluster members, and specify properties for the cluster.

A cache cluster uses one configuration that is propagated from the current cache (the cache to which your client 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
- Auto-restart
- Application Web servers
- Proxy servers
- Sites
- Caching and expiration rules
- Error pages
- Session management

The following settings are specific to each member of the cluster:

- Process identity
- Network timeouts

- Resource limits

- End-User Performance Monitoring

- Event logs

- Access logs

- Operations ports, such as administration, invalidation, and statistics ports

- Listener ports

- Origin server wallet

This section contains the following topics to help you in configuring a cache cluster:

- Configuration Prerequisites

- Understanding Failover Threshold and Capacity Settings

- Task 1: Configure Cache Cluster Settings

- Task 2: Add Caches to the Cluster

- Task 3: Enable Tracking of Session Binding

- Task 4: Propagate the Configuration to Cluster Members

In addition, see the following information about configuring clusters:

- Removing Caches from a Cluster

- Configuring Administration and Invalidation-Only Clusters

> **See Also:** Chapter 3 for an overview of cache clusters

Configuration Prerequisites

Because a cache cluster contains two or more instances of OracleAS Web Cache, you must have two or more instances of OracleAS Web Cache installed on one or more nodes before you configure a cache cluster. The instances must be the same version of OracleAS Web Cache. In addition, the respective passwords for the OracleAS Web Cache administrator, `administrator`, and the invalidator user, `invalidator`, must be the same across the cluster members.

You can modify the passwords in the Security page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Security**) or OracleAS Web Cache Manager (**Properties** > **Security**).

Understanding Failover Threshold and Capacity Settings

To ease with configuration, take the time to understand the following key configuration settings for a cache cluster and its members:

- Failover Threshold for the Cache Cluster

- Capacity for Cache Cluster Members

Failover Threshold for the Cache Cluster

You set the failover threshold when you configure cache cluster properties. This setting reflects the number of allowed consecutive request failures before OracleAS Web Cache considers another cache cluster member to have failed. OracleAS 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 one of the following: `500 Internal Server Error`, `502 Bad Gateway`, `503 Service Unavailable`, or `504 Gateway Timeout`.

For each failed request, OracleAS 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, OracleAS Web Cache resets the failure counter.

When the failover threshold is met, OracleAS Web Cache considers the cache cluster member to have failed. OracleAS 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, OracleAS Web Cache starts polling the cache cluster member. It does this by sending requests to the ping URL you specify. When OracleAS 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.

Capacity for Cache Cluster Members

When you configure a cache cluster member, you specify capacity for that member.

OracleAS Web Cache uses capacity 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 served 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, OracleAS 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, OracleAS 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.

Set the initial capacity for each cache cluster member to 10 percent of the maximum number of incoming connections. You can find this setting in the Resource Limits and Timeouts page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Resource Limits and Timeouts**) or the Resource Limits page of OracleAS Web Cache Manager (**Properties** > **Resource Limits**.

Once you have a better idea of the site's capacity needs and hit rates, fine tune the capacity. If these two assumptions apply to your cache cluster, then apply the following formula to determine the capacity for each cluster member:

1. Incoming traffic will be distributed equally to all the cache cluster members.

2. Ownership of content will be distributed equally among all the cache cluster members.

In the following formula, pick the highest value between the default value or the `max_incoming_connections` formula:

```
max(default_value, (max_incoming_connections * (cacheable_misses%/100) * (number_of_caches - 1) / number_of_caches))
```

In the formula:

- `default_value` is one of the following:
  - 100 for production environments
  - 30 for test environments
  - 0 for invalidation-only clusters

  When the capacity increases, the number of file descriptors needed by OracleAS Web Cache also increases.

  > **See Also:**
  >
  > - "Configuring Administration and Invalidation-Only Clusters" on page 10-10 for further information about invalidation-only clusters
  > - "Problem 5: More Than 1,024 Connections" on page E-4 for further information about how to increase the number of file descriptors

- `max_incoming_connections` is the maximum number of incoming connections.

  You can find this setting in the Resource Limits and Timeouts page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Resource Limits and Timeouts**) or the Resource Limits page of OracleAS Web Cache Manager (**Properties** > **Resource Limits**).

- `cacheable_misses%` is the percentage of requests for cacheable objects that were not served directly by OracleAS Web Cache, but were served by OracleAS Web Cache after it fetched the content from the origin server.

  You can find the **Cacheable Misses** setting in the Performance page of Application Server Control Console (**Web Cache Home** page > **Performance** tab) or the Web Cache Statistics page of OracleAS Web Cache Manager (**Monitoring** > **Web Cache Statistics**). Use the statistic reported in the **Average (per second)** column of the Performance page and the **Since Start** column of the Web Cache Statistics page, respectively.

For example, assume a cache cluster with four members. If OracleAS Web Cache is operating at 1500 maximum incoming connections, with a 30 percent cacheable miss rate, then the equation to calculate capacity for this configuration looks like the following:

```
(1500 * (30/100) * (4 - 1) / 4
```

The equation calculates to 337.5. You would round up to 338, which is the capacity you would then enter for each cache cluster member.

```
1500 * .3 * 3 / 4 = 337.5
```

If you assign a capacity of 0 to a cluster member, that cluster member will not receive requests from other cluster members. However, that cluster member will forward requests to other cluster members, the owners of the content. If you assign a capacity of 0 to *all* cluster members, no requests will be forwarded between cluster members. Even when capacity is set to 0, you can still propagate the configuration and OracleAS Web Cache can automatically propagate invalidation requests to cluster members.

> **See Also:** "Configuring Administration and Invalidation-Only Clusters" on page 10-10 for more information

Task 1: Configure Cache Cluster Settings

To configure settings for a cache cluster in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Cluster Properties** > **Members and Properties**.

> **See Also:** "Configuring Cache Cluster Settings" in Enterprise Manager Online Help for instructions

To configure settings for a cache cluster in OracleAS Web Cache Manager:

1. In the navigator frame, select **Properties** > **Clustering**.

   The Clustering 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. OracleAS Web Cache ignores the cluster information if there is only one cluster member.

2. In the **General Cluster Information** section of the Clustering page, click **Edit**.

   The **Edit 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 OracleAS Web Cache considers another cache cluster member to have failed.

   The default is five failures.

   See "Failover Threshold for the Cache Cluster" on page 10-2 for further information about this field.

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 is cacheable and that you can guarantee is stored in each cache. The default is `_oracle_http_server_webcache_static_.html`, which is stored in the cache.

6. In the **Ping Interval** field, enter the time, in seconds, between attempts by a cluster member to reach the failed cluster member.

    The default, 10 seconds, is a reasonable interval for most situations.

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. Click **Submit.**

9. In the Cluster Members table of the Clustering page, default values are displayed for the current cache. Select the cache and click **Edit Selected.**

    The Edit Cluster Member dialog box appears.

10. In the **Cache Name** field, enter a name for the OracleAS 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 OracleAS 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 OracleAS 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 OracleAS Web Cache can sustain.

    See "Capacity for Cache Cluster Members" on page 10-3 for further information about this field.

14. Click **Submit.**

    You now have one cache, the current cache, in the cluster. However, the cluster information is ignored until you have more than one OracleAS 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: Restart OracleAS Web Cache" on page 8-42 for information about starting OracleAS 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 8-7.

To add another cache to the cluster in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Cluster Properties** > **Members and Properties**.

> **See Also:**   "Adding a Cache to a Cluster" in Enterprise Manager Online Help for instructions

To add another cache to the cluster in OracleAS Web Cache Manager:

1. In the navigator frame, select **Properties** > **Clustering**.

    The Clustering page appears.

2. In the Cluster Members section of the Clustering page, click **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 **Protocol for Admin Port** field, select either **HTTP** or **HTTPS** to accept HTTP or HTTPS client requests.

6. 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.

7. In the **Capacity** field, enter the number of concurrent incoming connections from other cache cluster members that OracleAS Web Cache can sustain.

   > **See Also:** Step 13 in "Task 1: Configure Cache Cluster Settings" on page 10-5 for more information about capacity

8. Click **Submit.**

   The cache is now part of the cluster and is listed in the Cluster Member table.

9. To add more OracleAS Web Cache instances to the cache cluster, repeat Steps 2 through 8.

10. When you have completed adding members to the cache cluster, click **Apply Changes**.

OracleAS Web Cache adds the cache-specific information from the new cache cluster members to the cluster configuration.

You can add more OracleAS 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: Enable Tracking of Session Binding

In a cache cluster, all cache cluster members must be able to determine which origin server established the session, although the request was routed originally through only one cache cluster member. To configure **session binding** in a cache cluster, you select a session binding mechanism of **Cookie-based**. Setting this mechanism adds a cookie that tracks session information so that it can be read by all cluster members. OracleAS Web Cache includes a `Set-Cookie` response-header in the response so that subsequent requests from the client include the cookie. The cookie provides information so that any of the cluster members can resolve the binding regardless of which cache handled the initial request.

> **See Also:**
>
> - "Session Binding (Stateful Load Balancing)" on page 1-16 for an overview of session binding
>
> - "Bind a Session to an Origin Server" on page 8-39 for instructions on enabling session binding and tracking session binding in a cache cluster

Task 4: Propagate the Configuration to Cluster Members

When you modify the cluster and apply changes, OracleAS 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 in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Cluster Properties** > **Cluster Operations**.

> **See Also:** "Propagate the Configuration to Cluster Members" in Enterprise Manager Online Help for instructions

To propagate the configuration to new cluster members in OracleAS Web Cache Manager:

1. In the navigator frame, select **Operations** > **Cache Operations**.

    The Cache Operations page appears. The **Operation Needed** column indicates the caches to which the configuration should be propagated.

2. Propagate the configuration to all cache cluster members:

    a. Select **All caches** in the **Operate On** field.

    b. Select an **Interval** of **Immediate**. (No other interval is allowed for propagation.)

    c. Click **Propagate**.

    (Alternatively, you can propagate the configuration to one cluster member at a time. Click **Selected cache** in the **Operate On** field, and then click **Propagate**.)

    When the operation completes, the **Operation Needed** column in the Cache Operations page indicates the cluster members that need to be restarted.

3. 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 click **Restart.**

    (Alternatively, you can restart one cluster member at a time.) Choose **Selected cache** in the **Operate On** field and then click **Restart.**)

When the operation completes, the **Operation Needed** column in the Cache Operations page indicates that no operations are needed. The cache cluster is ready to use.

Removing Caches from a Cluster

To remove a cache from a cluster, you must not only make sure that remaining cluster members no longer include that cache in cluster, but that the removed cache no longer considers itself to be part of the cluster.

To remove a cache from a cluster in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Cluster Properties** > **Members and Properties**.

> **See Also:** "Removing a Cache from a Cluster" in Enterprise Manager Online Help for instructions

The script `chgiphost.sh` enables you to change the host name or IP address of a computer. If the computer contains a middle-tier instance that is part of OracleAS Web Cache cluster, you must remove the instance from the cache prior to running this script.

> **See Also:** *Oracle Application Server Administrator's Guide* for more information about using the `chgiphost.sh` script

To remove a cache from a cluster in OracleAS Web Cache Manager:

1. Enter the URL for the OracleAS Web Cache Manager of one of the caches in cluster, but *not* the cache that you want to remove from the cluster. If prompted, enter the user name and password for the `ias_admin` or `administrator` user.

2. In the navigator frame, select **Properties** > **Clustering**.

3. In the Cluster Members section of the Clustering page, select the cache you want to remove from the cluster and click **Delete Selected.**

4. In the OracleAS Web Cache Manager main window, click **Apply Changes**.

5. Propagate the change to the other remaining cache cluster members:

   a. In the navigator frame, select **Operations > Cache Operations.**

   b. Select **All caches** in the **Operate On** field.

   c. Select an **Interval** of **Immediate**.

   d. Click **Propagate**.

   The change is propagated to all the remaining cluster members, but not to the removed cluster member.

6. Restart all cluster members:

   a. In the Cache Operations page, select **All caches** in the **Operate On** field.

   b. Select an **Interval** to stagger the time that operation begins on the caches, and then click **Restart.**

   All remaining caches in the cluster no longer consider the removed cache to be part of the cluster. However, the removed cache still considers itself to be part of the cluster. To remedy that situation, take the next steps.

7. Enter the URL for the OracleAS Web Cache Manager of the cache you removed from the cluster. Enter the username and password for the `ias_admin` or `administrator` user.

8. In the navigator frame, select **Properties** > **Clustering**.

   The Clustering page appears. The Cluster Members section still shows all members of the cluster.

9. In the Cluster Members section of the Clustering page, select each cache except the current one, and click **Delete Selected**. Repeat until only the current cache remains in the Cluster Members list.

10. In the OracleAS Web Cache Manager main window, click **Apply Changes**.

11. In the navigator frame, select **Operations > Cache Operations**.

12. Select the cache and click **Restart**.

Configuring Administration and Invalidation-Only Clusters

You can configure a cluster that supports propagating the configuration and invalidation requests across all cache cluster members, but that does not forward requests between cache cluster members. That is, in processing requests, each cluster member acts as an individual cache and does not request objects from its peer cluster members. However, configuration changes and invalidation requests can be propagated to cluster members.

You can use this configuration to simplify administration of many caches. It may be needed in a cluster where members are separated by a firewall. For example, you can have a cluster where two caches are located on either side of a firewall that separates the intranet from Internet. (The network settings of such a setup—of sending Internet traffic to one cache and intranet traffic to another—is beyond the scope of this document.)

To manage these caches as a cluster and avoid sharing contents between the caches, take the following steps:

1. Create a cluster and add members to it as discussed in "Task 1: Configure Cache Cluster Settings" on page 10-5 and "Task 2: Add Caches to the Cluster" on page 10-6, with the exception noted in the following step.

2. For each cluster member, set the capacity to 0. (Select **Properties** > **Clustering**. Then, select a cluster member and click **Edit**. In the Edit Cluster Member dialog box, set the **Capacity** to 0.)

3. Propagate the configuration to all cluster members, as described in "Task 4: Propagate the Configuration to Cluster Members" on page 10-8.

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 OracleAS 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.

OracleAS 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 in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Cluster Properties** > **Cluster Operations**.

> **See Also:** "Propagate the Configuration to Cluster Members" in Enterprise Manager Online Help for instructions

To check that all cluster members are using the same configuration and to propagate the configuration, if necessary, in OracleAS Web Cache Manager:

1. In the navigation pane, select **Operations** > **Cache Operations**.

   The Cache 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 Cache 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 click **Propagate**. (Alternatively, to operate on all caches in the cluster, in the **Operate On** field, choose **All caches**. For the **Interval**, select **Immediate**, and then click **Propagate**.)

   OracleAS Web Cache propagates the configuration from the current cache to the selected cluster member. When the operation completes, the **Operation Needed** column in the Cache Operations page indicates that the cache needs to be restarted.

4. Restart all cluster members, either individually or all caches:

   ■ To restart one cluster member, select the cache. Then, in the **Operate On** field, choose **Selected cache** and click **Restart.**

   ■ To restart 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 click **Restart**.)

11

Configuring a Hierarchy of Caches

This section describes additional configuration options available for **cache hierarchy** deployments. This section contains the following topics:

- Configuring a Distributed Cache Hierarchy
- Configuring an ESI Cache Hierarchy
- Additional Hierarchy Configuration for a Cache Cluster

> **See Also:**
>
> - "Cache Hierarchies" on page 1-10 for an overview of cache hierarchies
> - "Compression" on page 1-18 for a description of how compression works in a 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 other words, the central cache acts as an origin server to the remote cache.

To configure a distributed cache hierarchy, perform the tasks in "Tasks for Setting Up OracleAS Web Cache" on page 8-6 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 from the Origin Servers page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Origin Servers**) or OracleAS Web Cache Manager (**Origin Servers, Sites, and Load Balancing** > **Origin Servers**).

   - For the remote caches, configure the central cache as the origin server in the Origin Server page.

2. Create the same site definition for both the central and remote caches from the Sites page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**) or the Site Definitions page of OracleAS Web Cache Manager (**Origin Servers, Sites, and Load Balancing** > **Site Definitions**).

3. For both central and remote caches, map the site definition to the origin server (configured in Step 1) from the Sites page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**)

or the Site-to-Server Mapping page of OracleAS Web Cache Manager (**Origin Servers, Sites, and Load Balancing** > **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.

4. Route that network so that client browser requests are forwarded to the nearest cache.

5. Optionally, ensure that the ClientIP address information that is forwarded to the cache and origin server is valid.

> **See Also:** "Ensuring That ClientIP Headers Are Valid" on page 8-47

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: For automatic propagation of invalidation messages, OracleAS Web Cache passes the encoded `invalidator` password in the page request between the remote and central cache during the hierarchy registration process. This HTTP traffic is susceptible to network sniffing. If the network is unprotected and insecure, configure HTTPS ports as follows:

- In the Ports page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**) or the Listen Ports page of OracleAS Web Cache Manager (**Ports** > **Listen Ports**) of the central cache, disable the default HTTP port and configure an HTTPS port in its place.

- In the Ports page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**) or the Operations Ports page of OracleAS Web Cache Manager (**Ports** > **Operations Ports**) of the remote cache, disable the default HTTP port for invalidation and configure an HTTPS port in its place.

Table 11–1 on page 11-3 shows the example settings for the deployment depicted in "Deploying a Distributed Cache Hierarchy" on page 5-3.

Table 11–1 Settings for us.webche-host and jp.webche-host

Central Cache us.webche1-host and us.webche2-host	Remote Cache jp.webche-host	Setting Location in Oracle Enterprise Manager 10*g* Application Server Control Console and OracleAS Web Cache Manager
Port: `7777`	Port: `7777`	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports** OracleAS Web Cache Manager: **Ports** > **Listen Ports**
Host and Port: `app1-host1: 7778` Host and Port: `app1-host2: 7778` Host and Port: `app2-host: 7778`	Host and Port: `us.webche1-host: 7777` Host and Port: `us.webche2-host: 7777`	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Origin Servers** OracleAS Web Cache Manager: **Origin Servers, Sites, and Load Balancing** > **Origin Servers**
Host and Port: `www.app1.company.com:80` Host and Port: `www.app2.company.com:80`	Host and Port: `www.app1.company.com:80` Host and Port: `www.app2.company.com:80`	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**

Site Host and Port: `www.app1.company.com:80`
 	
Origin Server Host and Port: `app1-host1:7778`
 	
`app1-host2:7778`
 	
Site Host and Port: `www.app2.company.com:80`
 	
Origin Server Host and Port: `app2-host:7778`
	Site Host and Port: `www.app1.company.com:80`
	
	`www.app2.company.com:80`
	
	Origin Server Host and Port: `us.webche1-host:7777`
	
	`us.webche2-host:7777`
		Application Server Control Console:
		
		Web Cache Home page > **Administration** tab > **Properties** > **Application** > **Sites**
		
		OracleAS Web Cache Manager:
		
		Origin Servers, Sites, and Load Balancing > **Site-to-Server Mappings**

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 other words, the provider cache acts as an origin server to the subscriber cache.

To configure an ESI cache hierarchy, perform the tasks in "Tasks for Setting Up OracleAS Web Cache" on page 8-6 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 from the Origin Servers page of Enterprise Manager (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Origin Servers**) or OracleAS Web Cache Manager (**Origin Servers, Sites, and Load Balancing** > **Origin Servers**).

   ■ For the subscriber cache, configure the origin servers of the local site and the provider caches in the Origin Servers page.

2. Create site definitions:

■ For each provider cache, create a site definition for the ESI provider site from the Sites page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**) or the Site Definitions page of OracleAS Web Cache Manager (**Origin Servers, Sites, and Load Balancing** > **Site Definitions**).

■ 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 OracleAS 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) from the Sites page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**) or the Site-to-Server Mapping page of OracleAS Web Cache Manager (**Origin Servers, Sites, and Load Balancing** > **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

4. Optionally, ensure that the ClientIP address information that is forwarded to the cache and origin server is valid.

> **See Also:** "Ensuring That ClientIP Headers Are Valid" on page 8-47

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:** For automatic propagation of invalidation messages, OracleAS Web Cache passes the encoded `invalidator` password in the page request between the subscriber and provider cache during the hierarchy registration process. This HTTP traffic is susceptible to network sniffing. If the network is unprotected and insecure, configure HTTPS ports as follows:
>
> **1.** In the Ports page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**) or the Listen Ports page of OracleAS Web Cache Manager (**Ports** > **Listen Ports**) of the provider cache, disable the default HTTP port and configure an HTTPS port in its place.
>
> **2.** In the Ports page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**) or the Operations Ports page of OracleAS Web Cache Manager (**Ports** > **Operations Ports**) of the subscriber cache, disable the default HTTP port for invalidation and configure an HTTPS port in its place.

Figure 11–1 shows how to deploy support for multiple internal ESI provider sites.

Figure 11–1 Deploying Support for Multiple Internal ESI Provider Sites
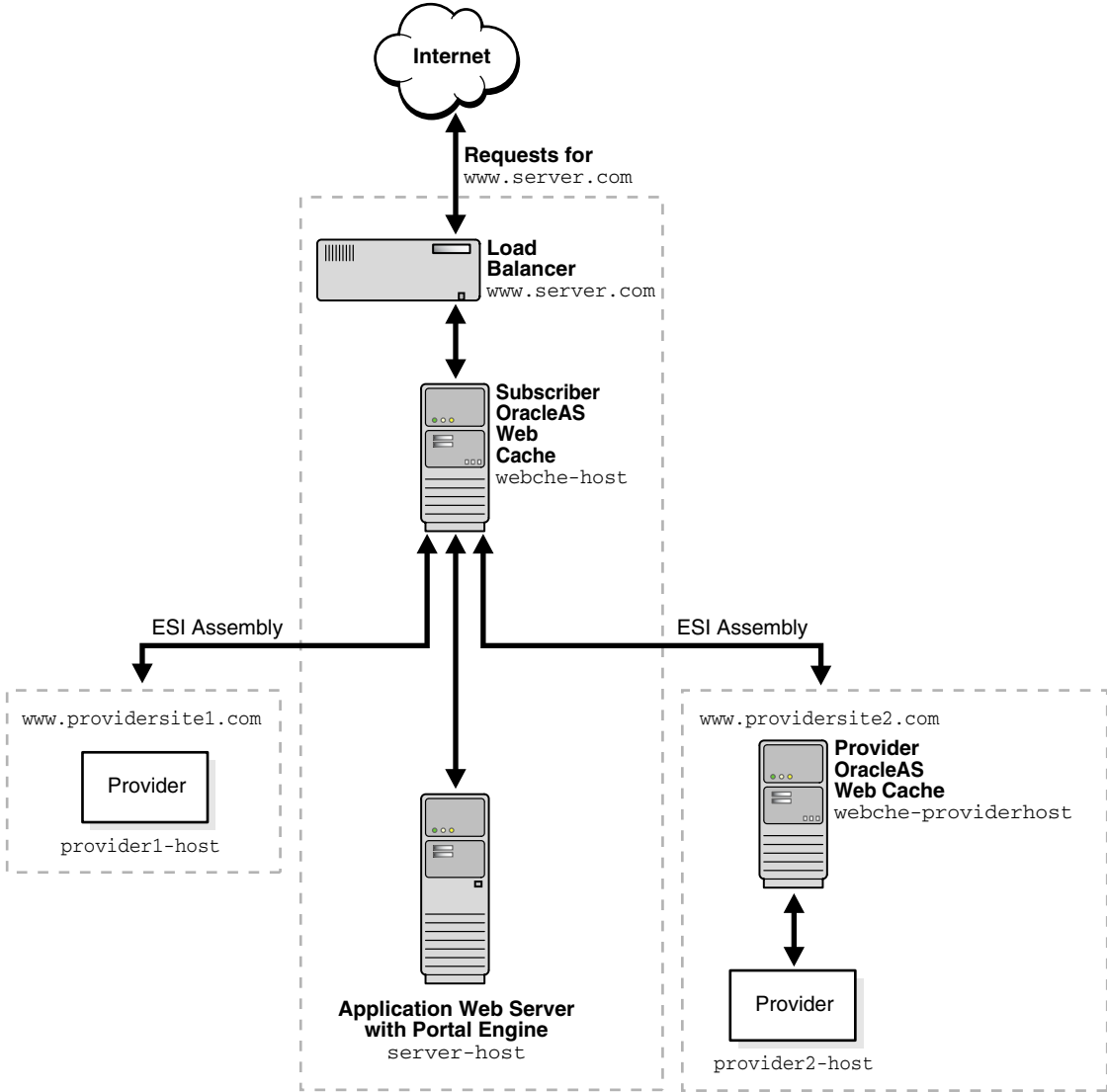
Table 11–2 on page 11-6 shows the example settings for the deployment depicted in Figure 11–1.

Table 11–2   Settings for webche1 and webche2

Subscriber Cache webche-host	Provider Cache webche-providerhost	Setting Location in OracleAS Web Cache Manager
Port: `7777`	Port: `7777`	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports** OracleAS Web Cache Manager: **Ports** > **Listen Ports**
Host and Port: `server-host: 7778` Host and Port: `provider1-host:7778` Host and Port: `webche-providerhost:7777`	Host and Port: `provider2-host: 7778`	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Origin Servers** OracleAS Web Cache Manager: **Origin Servers, Sites, and Load Balancing** > **Origin Servers**
Host and Port: `www.server.com:80` Host and Port: `www.providersite1.com:80` Host and Port: `www.providersite2.com:80`	Host and Port: `www.providersite2.com:80`	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites** OracleAS Web Cache Manager: **Origin Servers, Sites, and Load Balancing** > **Site Definitions**
Site Host and Port: `www.server.com:80` Origin Server Host and Port: `server-host:7778` Site Host and Port: `www.providersite1.com:80` Origin Server Host and Port: `provider1-host:7778` Site Host and Port: `www.providersite2.com:80` Origin Server Host and Port: `webche-providerhost:7777`	Site Host Name and Port: `www.providersite2.com:80` Origin Server Host and Port: `provider2-host:7778`	Application Server Control Console: **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites** OracleAS Web Cache Manager: **Origin Servers, Sites, and Load Balancing** > **Site-to-Server Mappings**

Additional Hierarchy Configuration for a Cache Cluster

In a cache hierarchy, each cache cluster member acts as an independent cache. Invalidation messages are propagated from each central or provider cache cluster member to the respective remote or subscriber caches. This configuration can result in the same invalidation message being propagated multiple times by cache cluster members.

If your deployment uses a load balancer to distribute requests among cache cluster members, you can optionally configure the cache cluster members to act as one cache and to receive only one set of propagated invalidation messages.

To configure cache cluster members to act as one cache for the purposes of hierarchical caching:

1. Select a cache cluster member to represent the cache.

2. For each of the other cache cluster members, use a text editor to open the `webcache.xml` file.

3. Locate the `CLUSTERINFO` element.

4. Add the following subelements:

```
<CLUSTERINFO NAME="CLUSTER_IP" VALUE="IP_address" />
<CLUSTERINFO NAME="CLUSTER_NORM_PORT" VALUE="web_cache_listening_Port" />
<CLUSTERINFO NAME="CLUSTER_INV_PORT" VALUE="web_cache_invalidation_port" />
<CLUSTERINFO NAME="CLUSTER_INV_SSL" VALUE="SSL_version" />
```

Table 11–3 describes how to enter values for the subelements.

Table 11–3   CLUSTERINFO Subelements

Subelement	Description
CLUSTER_IP	Enter the IP address of the cache.
CLUSTER_NORM_PORT	Enter the listening port of the cache. **See Also:** "Task 6: Configure OracleAS Web Cache with Listening Ports for Client Requests" on page 8-19
CLUSTER_INV_PORT	Enter the invalidation port of the cache. **See Also:** "Task 8: Configure OracleAS Web Cache with Operations Ports" on page 8-22
CLUSTER_INV_SSL	If the invalidation port is configured for HTTPS, configure the correct SSL version: ■ `NONE` for HTTP-only requests (default) ■ `SSL` for HTTPS requests supporting SSL version 1.0 ■ `SSLV2` for HTTPS requests supporting SSL version 2.0 ■ `SSLV3` for HTTPS requests supporting SSL version 3.0 ■ `SSLV3_V2H` for HTTPS requests supporting either SSL version 2.0 or 3.0

For example, if a cache cluster has members `webche1-host`, `webche2-host`, and `webche3-host`, you cans configure `webche2-host`, and `webche3-host` with settings for `webche1-host` as follows:

```
<CLUSTERINFO NAME="CLUSTER_IP" VALUE="130.35.45.41" />
<CLUSTERINFO NAME="CLUSTER_NORM_PORT" VALUE="9400" />
<CLUSTERINFO NAME="CLUSTER_INV_PORT" VALUE="9401" />
<CLUSTERINFO NAME="CLUSTER_INV_SSL" VALUE="NONE" />
```

This configuration enables the cache cluster to acts as one logic cache and only `webche1-host` to receive the propagated invalidation messages.

5. Save `webcache.xml`.

6. Restart OracleAS Web Cache with the following command:

```
opmnctl restartproc ias-component=WebCache
```

12

Creating Caching Rules

This chapter explains how to configure caching rules.

This chapter contains these topics:

- About Caching Rules
- Rule Creation
- Configuring Caching Rules and Rule Association
- Additional Configuration for Cache Policy Features
- Using the Surrogate-Control Response Header as an Alternative to Caching Rules
- Configuring Rules for Content Assembly and Partial Page Caching

About Caching Rules

You can choose to cache or not to cache content for static objects, multiple-version objects, personalized pages, pages that support a **session cookie**, **embedded URL parameter**, or **POST body parameter**, and dynamic pages with caching rules.

You configure a caching rule by specifying caching attributes based on the URL with Oracle Enterprise Manager 10*g* Application Server Control Console or OracleAS Web Cache Manager, or you configure the caching attributes for a specific object within a `Surrogate-Control` response-header field. Those objects contained within the URL are not cached until there is a client request for them. When an object is first requested, OracleAS Web Cache appends a `Surrogate-Capability` request-header field to the object. The `Surrogate-Capability` request-header field identifies that the object passed through the cache.

OracleAS Web Cache uses the following priority to determine object cacheability:

1. `Surrogate-Control` response header

2. Caching rule configured with Application Server Control Console or OracleAS Web Cache Manager

3. Other HTTP headers:

   - `Authorization` request header
   - `Proxy-Authorization` request header
   - `Pragma: no-cache` response header
   - `Warning` response header

   If any of these headers are present, then OracleAS Web Cache does not cache the object.

4. Cookie values from `Cookie` request header and `Set-Cookie` response header

5. `Cache-Control` response header

6. `Expires` response header

The `Surrogate-Control` response-header field enables the origin server to override the caching rules configured through Application Server Control Console or OracleAS Web Cache Manager. When both a `Surrogate-Control` response header and a caching rule for the same object are present, OracleAS Web Cache merges the two. For example, if there is a caching rule for an non-cacheable object set in Application Server Control Console or OracleAS Web Cache Manager with compression enabled, and the response header contains `Surrogate-Control: max-age=30+60`, then OracleAS Web Cache respects both settings. OracleAS Web Cache uses the `max-age` control directive from the `Surrogate-Control` response-header to cache the object and the compression setting from the caching rule. If there is a conflict between the `Surrogate-Control` response header and a caching rule, then OracleAS Web Cache uses the settings from the `Surrogate-Control` response header.

If no caching rules or the `Surrogate-Control` response header are specified, then OracleAS 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.

> **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 OracleAS Web Cache, all objects are cleared from the cache. In addition, all statistics are cleared.

> **See Also:** "About Cache Population" on page 2-1 for a description of how OracleAS Web Cache determines cache population

Rule Creation

To create a caching rule, you specify the following:

- Selectors
- Caching Policy
- Cache-Key Policy
- Priority

Selectors

When OracleAS Web Cache receives a client request, it uses **selectors** to filter through the caching rules to locate the appropriate rule for the request. During caching rule creation, you specify the following selectors:

- Expression type

  The expression type is one of the following:

- File Extension Expression Type

- Path Prefix Expression Type

- Regular Expression

- URL expression

  Specify the URL based on the expression type.

- **HTTP request method**

  Specify the GET, GET with query string, or POST HTTP request methods of the objects.

- URL and POST body parameters

  Specify any embedded URL parameters or embedded POST body parameters.

- POST body expressions

  If the POST HTTP request method is selected, specify the HTTP POST body in regular expression syntax.

File Extension Expression Type

A file extension specifies a shared file extension type, such as gif. To specify a file extension, you simply enter the extension. Because OracleAS Web Cache internally starts the file extension with a period (.), it is not necessary to enter it. For example, both gif and .gif are valid entries.

Path Prefix Expression Type

A path prefix traverses the directory structure to locate the objects. Because OracleAS Web Cache internally starts the path with http://*host_name:port/* or "/", it is not necessary to enter this information.

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 (\).

For example, a path prefix of http://www.company.com/contacts or /contacts matches objects under the contacts directory for the defined site www.company.com.

Regular Expression

OracleAS Web Cache supports **regular expression** syntax 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 12–1 shows examples of content to cache and how to enter regular expression syntax for corresponding caching rules for that content.

Table 12–1   Regular Expression Examples

Content to Cache	Regular Expression Syntax
URL beginning with /machine/doc and ending in *.gif	^/machine/doc/.*\.gif$
/robots.txt file	^/robots.txt$
All procedures in the new_employee package	^/pls/enroll_db/new_employee

Caching Policy

A caching policy specifies whether an object should be cached or not cached. A caching policy of **Cache** instructs OracleAS Web Cache to serve requests from objects in its cache; a caching policy of **Don't Cache** instructs OracleAS Web Cache to forward requests to the origin server and to not cache the content.

Examples of content that administrators typically declare non-cacheable include updating transactions, shopping cart views, and personal account views. One of the easiest ways to set up caching rules in OracleAS 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.

Cache-Key Policy

A cache-key policy enables OracleAS Web Cache to map requests to the appropriate objects in the cache. After OracleAS Web Cache uses the selectors and caching policy to filter requests, it uses the cache-key policy to locate the correct object in the cache.

OracleAS Web Cache composes a cache key with the following attributes that you specify:

- Site name
- URL
- HTTP methods
- POST body expression
- Cookie names and values for multiple-version objects
- HTTP request header names and values for multiple-version objects

Together, these attributes create a unique ID for a cached object, which OracleAS Web Cache then uses for request lookups.

Priority

You order the priority of caching rules. Higher priority rules are matched first.

When ordering caching rules for cacheable and non-cacheable objects, give the non-cacheable objects a higher priority than the cacheable objects. 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 with either regular expression or path prefix syntax, in the order shown in Table 12–2.

Table 12–2   Example of Priority for Non-Cacheable and Cacheable URLs

Priority	Expression Type	URL Expression	HTTP Method(s)	Caching Policy

1	Regular Expression
	Path Prefix
		`^/cec/cstage\?ecaction=ecpassthru2`
		
		`/cec/cstage\?ecaction=ecpassthru2`
			`GET`, `GET` with query string	Don't Cache

2	Regular Expression
	Path Prefix
		`^/cec/cstage\?ecaction=ecpassthru.*`
		
		`/cec/cstage\?ecaction=ecpassthru`
			`GET`, `GET` with query string	Cache

If the order were reversed, all objects 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.

In the rules shown in Table 12–3, Rule 2 caches objects of the URL that use the `GET` and `GET` with query string methods, and Rule 3 caches objects of the URL that use the `POST` method and a `POST` body matching `action=search`.

Table 12–3   Example of Priority for Different HTTP Methods

Priority	Expression Type	URL Expression	HTTP Method(s)	POST Body Expression	Caching Policy
1	Regular Expression Path Prefix	`^/cec/cstage\?ecaction=ecpassthru2` `^/cec/cstage\?ecaction=ecpassthru2`	`GET`,`GET` withquery string	N/A	Don't Cache
2	Regular Expression Path Prefix	`^/cec/cstage\?ecaction=ecpassthru.*` `/cec/cstage\?ecaction=ecpassthru`	`GET`,`GET` withquery string	N/A	Cache
3	Regular Expression Path Prefix	`^/cec/cstage\?ecaction=ecpassthru.*` `/cec/cstage\?ecaction=ecpassthru`	`POST`	`action=search`	Cache

Default Caching Rules

When OracleAS Web Cache is installed, site-specific and global caching rules are established for the configured default site.

Figure 12–1 on page 12-6 displays the Rules page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Rules**), and Figure 12–2 on page 12-6 displays the Caching, Personalization, and

Compression Rules page of OracleAS Web Cache Manager (**Rules for Caching, Personalization, and Compression** > **Caching, Personalization, and Compression Rules**).

Figure 12–1   Default Caching Rules in Application Server Control Console

Figure 12–2   Default Caching Rules in OracleAS Web Cache Manager

Table 12–4 describes the default caching rules. Rule 1 is intended for Oracle Application Server Wireless. If you are not using this Oracle Application Server component, you can delete this rule.

Table 12–4   Default Caching Rules

Priority	Expression Type	URL Expression	HTTP Method(s)	Cache/Don't Cache	Description
1	Regular Expression	`/ptg/rm`	GET, GET with query string	Cache	Caches the default Oracle Application Server Wireless servlet. This rule is necessary for Wireless to use OracleAS 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.
2	Regular Expression	`\.(gif|jpe?g|png|bmp)$`	GET	Cache	Caches all `.gif`, `.jpg`, `.jpeg`, `.png`, and `.bmp` files
3	File Extension	`.css`	GET	Cache	Caches `.css` (cascading style sheet) files
4	Regular Expression	`/jsLibs/.*\.js$`	GET	Cache	Caches all UIX Oracle JDeveloper `.js` (JavaScript) files without compression.
5	File Extension	`.js`	GET	Cache	Caches all `.js` files. Netscape has known issues with compressed JavaScript files.
6	Regular Expression	`\.html?$`	GET	Cache	Caches 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.
7	File Extension	`.swf`	GET	Cache	Caches all `.swf` files

> **See Also:** *Oracle Application Server Wireless Administrator's Guide* for further information about Oracle Application Server Wireless

Configuring Caching Rules and Rule Association

This section describes how to configure caching rules and associate the rules with multiple URLs. It contains the following topics:

- Task 1: Create Caching Rules

- Task 2: Prioritize Rules

- Task 3: (Optional) Associate Multiple Rules with Caching Policy Features

Task 1: Create Caching Rules

To create caching rules in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Rules**.

> **See Also:** "Configuring General Settings for Caching Rules" in Enterprise Manager Online Help for instructions

To create caching rules in OracleAS Web Cache Manager:

1. In the navigator frame, select **Rules for Caching, Personalization, and Compression** > **Caching, Personalization, and Compression Rules**.

   The Caching, Personalization, and Compression Rules page appears.

2. From the **For Site** list, select the Web site for which to view or create site-specific caching rules.

3. If no rules exist, choose **Create Site Specific Rule** or **Create Global Rule**. If rules already exist, then select a rule and choose **Insert Above** or **Insert Below**.

   The Edit/Add Caching, Personalization, and Compression Rule dialog box appears.

4. Specify the objects to which to apply the caching rule:

   a. From the **Expression Type** list, select one of the following options:

      * File Extension: Applies the caching rule to objects ending in a particular file extension, such as `.gif`

      * Path Prefix: Applies the caching rule to objects matching a path prefix

      * Regular Expression: Applies the caching rule to object matching **regular expression** syntax

   b. In the **URL Expression** field, specify the expression based on the selection you made for **Expression Type**:

      * File Extension: Enter the file extension. Because OracleAS Web Cache internally starts the file extension with a period (`.`), it is not necessary to enter it.

      * Path Prefix: Enter the path prefix of the objects. Because OracleAS Web Cache internally starts the path with `http://host_name:port/` or "/", it is not necessary to enter this information.

        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 (`\`).

      * Regular Expression: Enter the regular expression of the objects. Remember to use "`^`" to denote the start of the URL and "`$`" to denote the end of the URL.

        See Also: "Selectors" on page 12-2 for caching rule syntax

5. In the **HTTP Method(s)** section, select to cache objects 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, do not select **GET with query string** or **POST**. These options should only be selected if the forms are used in search forms.

6. For the File Extension and Path Prefix expression types, in the **URL and POST Body Parameters** section, enter any embedded URL parameters or `POST` body parameters and their values in the corresponding **Name** and optional **Value** fields. Then click **Add**.

For the Regular Expression expression type, you must alphabetically sort and enter the embedded URL parameters in the **URL Expression** field.

The request URL that client browsers send to OracleAS Web Cache and the internal URL expression that OracleAS Web Cache uses for that request are different. When OracleAS 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, you either use the **URL and POST Body Parameters** section or manually enter the embedded URL parameters alphabetically in regular expression syntax. When you use the **URL and POST Body Parameters** section, the embedded URL parameters are automatically sorted.

For example, consider the following URL:

```
http://my.oracle.com/servlet/page?_pageid=53&_dad=moc&_schema=MOC
```

If you enter the regular expression without manually sorting the embedded URL parameters in the **URL Expression** field, `^/servlet/page\?_pageid=53&_dad=moc&_schema=MOC$`, then the caching rule will not match the internal representation of the URL used by OracleAS Web Cache. To ensure matching, you must enter the regular expression in the **URL Expression** field as:

```
^/servlet/page\?_dad=moc&_pageid=53&_schema=MOC$
```

7. If you selected **POST** in the **HTTP Method(s)** section and did not specify POST body parameters in the **URL and POST Body Parameters** section, specify the HTTP POST body in the **POST Body Expression** field.

   To apply this rule to any POST request body, enter ".*" in the field.

8. In **Caching Policy**, select either **Cache** or **Don't Cache** for the objects matching the URL.

9. In **ESI Propagation Policy**, select either **Propagate** or **Don't Propagate**. The default is **Propagate**.

   - Select **Propagate** to enable **Edge Side Includes (ESI)**-compliant proxy caches to process ESI tags. Select this option if you prefer clients acting as caches to perform the ESI processing rather than OracleAS Web Cache.

   - Select **Don't Propagate** to disallow other ESI-compliant caches or services from processing ESI tags.

     See Also: "Configuring Rules for Content Assembly and Partial Page Caching" on page 12-33

10. Optionally, to help keep track of the meaning of rules, enter a comment for the caching rule in the **Comment** field.

11. In the **Compression** section:

    - Select **None** to not serve compressed cacheable and non-cacheable objects to browsers.

    - Select **For all browsers** to serve compressed objects to all browser types.

    - Select **For non-Netscape browsers only** to serve compressed objects for all browsers other than Netscape.

    Note that even if compression is enabled, OracleAS Web Cache does not compress the following:

- GIF, JPEG, PDF, PNG, SWF, and ZIP files

- Responses containing a `Content-Encoding` response-header field, which is typically used to denote compression

- Responses containing a `Content-Disposition` response-header field, which is typically used for attachments

   Responses with `Content-Disposition` response-header fields show incorrect file names when they are compressed.

- For certain browsers, JavaScript files

   Compressed JavaScript files cause some browsers to behave erratically and possibly fail. This issue only affects files that are referenced with the `src` attribute of the `script` tag; it does not include files that contain inline JavaScript.

   OracleAS Web Cache does not compress JavaScript files for Netscape 4.*x* and Internet Explorer 5.5 browsers. For other browsers, it compresses the file if compression is enabled.

- For certain browsers, cascading style sheets (`.css`)

   OracleAS Web Cache does not compress cascading style sheets for Netscape 4.*x* and Internet Explorer 5.5 browsers. With these browsers, compressed cascading style sheets can cause background attributes, such as background images, to not appear in the output.

   Oracle recommends not compressing executables and files that are already zipped with utilities like WinZip and GZIP. Compressing these files incurs additional overhead without the benefits of compression.

   > **See Also:** "Compression" on page 1-18 for an overview of compression

12. In the **Enabled** section, select **Yes** to enable the caching rule, or select **No** to disable the caching rule.

   Oracle recommends deselecting a caching rule for test systems

13. If you selected **Cache** in the **Cache Policy** section, select the options in Table 12–5 that apply, and then click **Submit**.

Table 12–5   Options for Rules with a Cache Policy of Cache

Option	Explanation
Expiration Policy	From the list, select an expiration policy to apply to the objects. If you do not see an expiration policy suitable for the objects, in the navigator frame, select **Rules for Caching, Personalization, and Compression** > **Expiration Policy Definitions** to create a new policy. **See Also:** - "Expiration" on page 2-2 for an overview - "Configuring Expiration Policies" on page 12-14 for additional configuration details

Table 12–5   (Cont.) Options for Rules with a Cache Policy of Cache

Option	Explanation
Multiple Objects with the Same Selector by Cookies	To cache multiple versions of an object that rely on **category cookie** values, select the required cookies. If you do not see a cookie that can be applied to these objects, in the navigator frame, select **Rules for Caching, Personalization, and Compression** > **Cookie Definitions** to create a new cookie definition.
	See Also:
	■ "Multiple Versions of the Same Object" on page 2-4 for an overview
	■ "Configuring Cookie Definitions for Multiple-Version Objects Containing Cookies" on page 12-15 for additional configuration details
Multiple Objects with the Same Selector by Other Headers	To cache multiple versions of an object that rely upon **HTTP request header** values, 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 client 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,*/*s
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
``` |
	Note: OracleAS Web Cache does not interpret the values of these HTTP request headers. If the values for two pages are different, OracleAS Web Cache caches both pages separately. This issue is especially problematic with the `User-Agent` request header, whereby the browser type, version, and operating system can result in multiple cache entries. You can override this behavior for the `User-Agent` request header by configuring OracleAS Web Cache to cache and serve the same page for the same browser type, as described in "Recognizing Similar Browser Types for Multiple-Version Objects Containing HTTP Request Headers" on page 12-16.
	See Also: "Multiple Versions of the Same Object" on page 2-4 for an overview
Session Caching Policies	To cache or serve objects based on session or **personalized attribute** information contained within a cookie, embedded URL parameter, or `POST` body parameter, select the sessions and corresponding policy. If you do not see a session that can be applied to these objects, in the navigator frame, select **Rules for Caching, Personalization, and Compression > Session Definitions** to create a new session.
	See Also:
	■ "Controlling How the Cache Serves Personalized Attribute Requests" on page 2-9 or "Controlling How Session Requests Are Served by the Cache" on page 2-12 for an overview
	■ "Configuring Session or Personalized Attribute Caching Policies" on page 12-19 for additional configuration details

Table 12–5 (Cont.) Options for Rules with a Cache Policy of Cache

Option	Explanation
Session-Encoded URL	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. OracleAS Web Cache will replace the value information based on the value of the cookie, embedded URL parameter, or `POST` body parameter. OracleAS Web Cache then serves these pages for client requests that contain the cookie or the parameter.
	See Also:
	■ "Substituting Session Information in Session-Encoded URLs" on page 2-11 or "Personalized Attributes" on page 2-7 for an overview
	■ "Configuring Rules for Popular Pages with Session Establishment" on page 12-29 for additional configuration details
Avoid Unwanted Copies	To define additional embedded URL or `POST` body parameters for which OracleAS Web Cache ignores the values, click **Global URL Parameters to Ignore** to apply the parameter to all caching rules or **Site-Specific URL Parameters to Ignore** for rules configured for this site.
	By configuring OracleAS Web Cache to ignore the value of embedded URL or `POST` body parameters, you enable OracleAS Web Cache to serve one cached object to multiple sessions requesting the same page. OracleAS Web Cache caches the response to the first request and serves subsequent requests for the page from its cache.
	See Also:
	■ "Excluding the Value of Embedded URL or POST Body Parameters" on page 2-10 for an overview
	■ "Configuring OracleAS Web Cache to Exclude the Value of Embedded URL or POST Body Parameters" on page 12-18 for additional configuration details
HTTP Error Caching	Enter the HTTP error codes you want OracleAS Web Cache to cache and serve for this rule. If you enter multiple codes, use a comma to separate them.
	If there is a problem on the **origin server** that does not result in a `200 OK` HTTP response status for this rule, then OracleAS Web Cache serves the cached HTTP error, saving origin server resources. The origin server must generate the HTTP error itself.
	OracleAS Web Cache caches the error pages according to the expiration policy of the rule.
	After the problem is resolved, invalidate the HTTP errors.
	See Also: Chapter 13, "Sending Invalidation Requests"

14. Repeat Steps 2 through 13 for each caching rule.

> **Tip:**   In addition to, or as an alternative to, creating caching rules with Application Server Control Console or OracleAS Web Cache Manager, application developers can choose to store the many of the caching attributes in the header of an HTTP response message. See "Using the Surrogate-Control Response Header as an Alternative to Caching Rules" on page 12-30 for details.

Task 2: Prioritize Rules

After you configure the caching rules, order them. Only the first matching rule is honored.

To order rules:

1. In the Caching, Personalization, and Compression Rules page, select a caching rule, and then choose **Move Up** or **Move Down** to order the rules.

2. Apply changes and restart OracleAS Web Cache:

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

   b. In the Cache Operations page, select a cache, and then click **Restart** to restart OracleAS Web Cache.

Task 3: (Optional) Associate Multiple Rules with Caching Policy Features

In OracleAS Web Cache Manager, you can associate a group of rules with the following cache policy features:

- ESI propagation

- Compression

- Expiration

- Cookies for multiple-version objects

- HTTP request headers for multiple-version objects

- Session caching

- Session-encoded URLs

- HTTP error caching

To associate multiple rules with a cache policy feature:

1. In the navigator frame, select **Rule Association >** *Policy* **Association.**

   The Association page for the cache policy feature appears.

2. In the Association page, select the policy, and then click **Change Rule Association.**

   The Change Association dialog box for the cache policy feature appears.

3. Select one or more caching rules in the right list, and then click **Make Association.**

   If the caching rule you require does not exist, create a caching rule, as described in "Configuring Caching Rules and Rule Association" on page 12-7.

4. Apply changes and restart OracleAS Web Cache:

   a. In the OracleAS Web Cache Manager main window, click **Apply Changes.**

   b. In the Cache Operations page, select a cache, and then click **Restart** to restart OracleAS Web Cache.

Additional Configuration for Cache Policy Features

This section contains the following topics about configuring cache policy features:

- Configuring Expiration Policies

- Configuring Cookie Definitions for Multiple-Version Objects Containing Cookies

- Recognizing Similar Browser Types for Multiple-Version Objects Containing HTTP Request Headers

- Configuring OracleAS Web Cache to Exclude the Value of Embedded URL or POST Body Parameters

- Configuring Session or Personalized Attribute Caching Policies

- Configuring Support for Session-Encoded URLs
- Configuring Support for Personalized Attributes
- Configuring Rules for Popular Pages with Session Establishment

Configuring Expiration Policies

You can create policies that specify when to expire objects in the cache. In addition, you can specify how long objects can reside in the cache after they have expired. When an object expires, OracleAS Web Cache removes it either immediately or as permitted by origin server capacity up to a maximum time limit.

To configuration expiration policies in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Rules** > **Expiration Policies**.

> **See Also:** "Configuring Expiration Policies" in Enterprise Manager Online Help for instructions

To create expiration policies in OracleAS Web Cache Manager:

1. In the navigator frame, select **Rules for Caching, Personalization, and Compression** > **Expiration Policy Definitions**.

   The Expiration Policy Definitions page appears.

2. In the Expiration Policy Definitions page, click **Add.**

   The Create Expiration Policy dialog box appears.

3. In the **Expire** section, specify when to expire objects by selecting one of the following options:

   - *<time>* **after cache entry:** Select this option to base expiration on when the objects entered the cache. Enter a number and select an interval, such as seconds, to expire the objects.

   - *<time>* **after object creation:** Select this option to base expiration on when the objects were created. Enter a number and select an interval, such as seconds, to expire the objects.

   - **as per HTTP Expires header:** Select this option to respect the HTTP `Expires` response-header field. This is the default. To use this option, objects must use the HTTP `Expires` or `Cache-Control` response-header fields.

   While the first two options enable you to set expiration for OracleAS Web Cache-specific rules, the third option recognizes the expiration policy established for the objects already programmed with an HTTP `Expires` or `Cache-Control` response-header field.

4. In the **After Expiration** section, specify how you want OracleAS Web Cache to process objects after they have expired:

   - **Remove immediately:** OracleAS Web Cache marks objects as invalid and then removes them immediately. An object is refreshed from the origin 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:** OracleAS Web Cache marks objects as invalid and then refreshes them based on origin server capacity. Enter the maximum time in which the objects can reside in the cache.

> **Note:** Performance assurance heuristics apply when you configure objects to be refreshed based on when the origin servers can refresh them; performance assurance heuristics do not apply to objects immediately removed.

5. Click **Submit**.

6. Repeat Steps 3 through 6 for each expiration policy.

7. In the navigator frame, select **Rule Association** > **Expiration Policy Association**.

The Expiration Policy Association page appears.

8. In the Expiration Policy Association page, select the newly-created policy, and then click **Change Rule Association**.

The Change Expiration Association dialog box appears.

9. Select a caching rule in the right list, and then click **Make Association**.

The caching rule moves to the left list and the dialog box closes.

If the caching rule you require does not exist, create a caching rule, as described in "Configuring Caching Rules and Rule Association" on page 12-7. In Step 13 of the procedure, select an expiration rule in the **Expiration Policy** row of the Edit/Add Caching, Personalization, and Compression Rule dialog box.

10. Apply changes and restart OracleAS Web Cache:

   a. In the OracleAS Web Cache Manager main window, click **Apply Changes**.

   b. In the Cache Operations page, select a cache, and then click **Restart** to restart OracleAS Web Cache.

Configuring Cookie Definitions for Multiple-Version Objects Containing Cookies

> **See Also:** "Multiple Versions of the Same Object" on page 2-4 for an overview and an example scenario

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

To specify cookie values for multiple-version URLs in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Rules**.

> **See Also:** "Configuring Advanced Settings for Caching Rules" in Enterprise Manager Online Help for instructions

To specify cookie values for multiple-version URLs in OracleAS Web Cache Manager:

1. In the navigator frame, select **Rules for Caching, Personalization, and Compression** > **Cookie Definitions**.

The Cookie Definitions page appears.

2. In the Cookie Definitions page, click **Add.**

The Edit/Add Cookie Definition dialog box appears.

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

4. For the prompt **Also cache objects whose requests do not contain this cookie?**, select either **Yes** or **No:**

   ■ Select **Yes** to cache versions of the object that do not contain this cookie. This option enables OracleAS Web Cache to serve objects from the cache for client requests that do not contain this cookie.

   ■ Select **No** to not cache versions of objects that do not contain this cookie.

5. Click **Submit.**

6. In the navigator frame, select **Rule Association** > **Cookie Association**.

   The Cookie Association page appears.

7. In the Cookie Association page, select the newly-created cookie definition, and then click **Change Rule Association.**

   The Change Cookie Association dialog box appears.

8. Select a caching rule in the right list, and then click **Make Association.**

   The caching rule moves to the left list and the dialog box closes.

   If the caching rule you require does not exist, create a caching rule, as described in "Configuring Caching Rules and Rule Association" on page 12-7. In Step 13 of the procedure, select a cookie in the **Multiple Objects with the Same Selector by Cookies** field of the Edit/Add Caching, Personalization, and Compression Rule dialog box.

9. Apply changes and restart OracleAS Web Cache:

   a. In the OracleAS Web Cache Manager main window, click **Apply Changes.**

   b. In the Cache Operations page, select a cache, and then click **Restart** to restart OracleAS Web Cache.

Recognizing Similar Browser Types for Multiple-Version Objects Containing HTTP Request Headers

By default, OracleAS Web Cache does not interpret the values of the HTTP request headers. When the **Multiple Objects with the Same Selector by Other Headers** for the `User-Agent` request-header field is selected in OracleAS Web Cache Manager and the value of the `User-Agent` request header of the same URL differ, then OracleAS 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)` and another request sends an HTTP request header of `User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows; DigExt)` for different versions of Internet Explorer, OracleAS Web Cache caches two separate pages.

You can override this default behavior by configuring OracleAS Web Cache with a `User-Agent` pattern string for a particular client. For the affected multiple-version objects, OracleAS Web Cache adds an `x-Oracle-Mapped-User` request-header field, and uses the value of the string rather than the entire `User-Agent` value:

`x-Oracle-Mapped-User: MAPPEDUSERAGENT_String`

> **See Also:** "Multiple Versions of the Same Object" on page 2-4 for an overview and an example scenario

To configure OracleAS Web Cache to cache and serve the same page for each browser type:

1. Create a caching rule for the pages that support the `User-Agent` request header, as described in "Configuring Caching Rules and Rule Association" on page 12-7. In Step 13 of the procedure, select `User-Agent` in the **Multiple Objects with the Same Selector by Other Headers** section of the Edit/Add Caching, Personalization, and Compression Rule dialog box.

2. Use a text editor to open the `webcache.xml` file.

3. Locate the `GLOBALCACHINGRULES` element.

4. For each browser type, add the following subelements in the next line after the `GLOBALCACHINGRULES` element:

```
<USERAGENTREMAPRULE MATCHSTRING="browser"
MAPPEDUSERAGENT="x-Oracle-Mapped-User-Agent_value" MAPTYPE="USERAGENT"/>
```

If you enter multiple entries, order them according to how you want OracleAS Web Cache to match. The order of these rules work in the same fashion as priority works for caching rules.

Table 12–6 describes how to enter values for the subelements.

Table 12–6   GLOBALCACHINGRULES Subelements

Subelement	Description
MATCHSTRING	Enter the pattern that will be used to match the incoming request header. **Note:** You can use the wildcard `*` to pattern match for multiple browser type variants. For example, `Mozilla*` can be used to match all variations of Mozilla.
MAPPEDUSERAGENT	Enter a unique value of the `User-Agent` pattern that will be added to the `x-Oracle-Mapped-User-Agent` request header by OracleAS Web Cache.
MAPTYPE	Enter `USERAGENT` to pattern match on the `User-Agent` request header.

The following `webcache.xml` fragment shows the `User-Agent` remapping:

```
<USERAGENTREMAPRULE MATCHSTRING="MSIE *" MAPPEDUSERAGENT="MSIE"
MAPTYPE="USERAGENT"/>
<USERAGENTREMAPRULE MATCHSTRING="Mozilla*" MAPPEDUSERAGENT="MOZ"
MAPTYPE="USERAGENT"/>
```

If an incoming request does not match any of the rules, OracleAS Web Cache appends a default mapping to the request. The default value of the `x-Oracle-Mapped-User-Agent` header is "DEFAULT_USER_AGENT."

These mapping rules are executed for every incoming request. If you create several mapping rules, you may experience a performance degradation.

5. Locate the `<MULTIVERSIONHEADERSRULE>` subelement of `CACHEABILITYRULE` for the caching rule created in Step 1.

```
<MULTIVERSIONHEADERSRULE>
     <HTTPHEADER NAME="User-Agent"/>
</MULTIVERSIONHEADERSRULE>
```

6. To match on the value of the `MAPPEDUSERAGENT` string rather than the entire `User-Agent` value, change the "User-Agent" header to "x-Oracle-Mapped-User-Agent" in the `HTTPHEADER` attribute of the rule:

```
<MULTIVERSIONHEADERSRULE>
     <HTTPHEADER NAME="x-Oracle-Mapped-User-Agent"/>
</MULTIVERSIONHEADERSRULE>
```

7. Save `webcache.xml`.

8. Restart OracleAS Web Cache with the following command:

   ```
   opmnctl restartproc ias-component=WebCache
   ```

Configuring OracleAS Web Cache to Exclude the Value of Embedded URL or POST Body Parameters

You can configure OracleAS Web Cache to ignore the value of embedded URL or POST body parameters. This configuration instructs OracleAS Web Cache to serve the same page to multiple sessions requesting the same page. OracleAS Web Cache caches the response to the first request and serves subsequent requests for the page from its cache.

> **See Also:** "Excluding the Value of Embedded URL or POST Body Parameters" on page 2-10 for an overview and an example scenario

You have three configuration options for specifying parameters to ignore. You can:

- Establish global URL parameters.

  - In Application Server Control Console, use the **Global URL Parameters to Ignore** link in the Sites page (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**). See "Excluding the Value of Embedded URL or POST Parameters " in Enterprise Manager Online Help.

  - In OracleAS Web Cache, navigate to the **Avoid Unwanted Copies** section in the Site Definitions page (**Origin Servers, Sites, and Load Balancing** > **Site Definitions**). See "Creating Site Definitions" on page 8-28.

- Specify site-specific parameters overrides.

  - In Application Server Control Console, select a configured site in the Sites page (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**), and click **Edit**. In the Create or Edit Named Site: Advanced tabbed page, navigate to the **URL Parameters to Ignore** section. See "Configuring Site Properties for a Named Site" in Enterprise Manager Online Help.

  - In OracleAS Web Cache Manager, select a configured site in the Site Definitions page (**Origin Servers, Sites, and Load Balancing** > **Site Definitions**), and then click **Edit Site**. See the online help for the Site Definitions page.

- When creating a caching rule, specify either global or site-specific parameters.

  - In Application Server Control Console, create a new caching rule in the Rules page (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Rules**). In the Create Rule: Advanced Caching Instructions tabbed page, navigate to the **Avoid Unwanted Copies** section. See "Configuring Advanced Settings for Caching Rules" in Enterprise Manager Online Help.

  - In OracleAS Web Cache Manager, create a new caching rule in the Caching, Personalization, and Compression Rules page (**Rules for Caching, Personalization, and Compression** > **Caching, Personalization, and**

Compression Rules). In the Edit/Add Caching, Personalization, and Compression Rules page, navigate to the **Avoid Unwanted Copies** section. See "Task 1: Create Caching Rules" on page 12-7.

Configuring Session or Personalized Attribute Caching Policies

You can specify how OracleAS Web Cache serves requests with the existence or nonexistence of session or personalized attribute cookies, embedded URL parameters, or POST parameters.

> **See Also:**
>
> - "Controlling How Session Requests Are Served by the Cache" on page 2-12 for an overview of caching rules for sessions
>
> - "Controlling How the Cache Serves Personalized Attribute Requests" on page 2-9 for an overview of caching rules for personalized attributes

To create caching policies for pages that support session cookies or personalized attributes in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Rules**.

> **See Also:** "Configuring Session or Personalized Attribute Caching Policies" in Enterprise Manager Online Help for instructions

To create caching policies for pages that support session cookies or personalized attributes in OracleAS Web Cache Manager:

1. In the navigator frame, select **Rule Association** > **Session Caching Policy Association**.

    The Session Caching Policy Association page appears.

2. In the Session Caching Policy Association page, click **Add.**

    The Add Session Caching Policy dialog box appears.

3. From the **Please select a session** list, select a session or personalized attribute, and then proceed to Step 5.

    If the sessions or personalized attributes listed do not contain the definition you require, then click **Cancel** to exit the Add Session Caching Policy dialog box. Continue to Step 5.

4. Create a session or personalized attribute definition:

    a. In the navigator frame, select **Rules for Caching, Personalization, and Compression > Session Definitions.**

    The Session Definitions page appears.

    b. From the **For Site** list, select the Web site for which to create site-specific definitions.

    c. Click **Add** or **Create.**

    The Edit/Add Session Definition dialog box appears.

    d. Select either **This Site Only** or **For All Sites.**

    e. In the **Session Name** field, enter an easy-to-remember unique name for the session or personalized attribute.

f. In the **Extract Value From** field, enter the cookie name in the **Cookie Name** field, or enter the embedded URL parameter or POST parameter in the **URL or POST body parameter** field.

If you enter both a cookie name and 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 client browser removes the cookie and subsequent requests for the object are directed to the origin server. To avoid pages from being served past the client session expiration time, ensure that the session cookie expires before the application Web server expires the client session.

g. In the **Default Value (Optional)** field, enter a default string that can be used for session-encoded URLs or personalized attributes.

> **See Also:**
>
> - Step 1g of "Configuring Support for Session-Encoded URLs" on page 12-21 for further information about this field for session-encoded URLs
>
> - Step 1g of "Configuring Support for Personalized Attributes" on page 12-23 for further information about this field for personalized attributes

h. Optionally, in the **Comment** field, enter a description of the definition.

i. Click **Submit.**

j. Repeats Steps 2 through 4.

5. Select **YES** or **NO** to the prompts in the Add Session Caching Policy dialog box:

> **Note:** With an embedded URL or POST body parameter, OracleAS Web Cache ignores the existence of an embedded URL parameter in the request. In other words, OracleAS Web Cache caches the response, even if the parameter in the request does not match the parameter in the response. To force the existence of the parameter in the client request, answer **YES** to the first prompt and **NO** to the second prompt.

a. For the prompt **1. Cache objects whose requests contain this session?**, select either **YES** or **NO**:

- Select **YES** to cache versions of objects that use the cookie or parameter.

- Select **NO** to not cache versions of objects that use the cookie or parameter.

b. For the prompt **2. Cache objects whose requests do NOT contain this session?**, select either **YES** or **NO**:

- Select **YES** to cache versions of objects that do not use the cookie or parameter. This selection enables OracleAS Web Cache to serve objects from the cache for client requests without the session or personalized attribute information.

- – Select **NO** to not cache versions of objects that do not use the cookie or parameter.

   c. If you answered **YES** to the prompts described in Steps 5a and 5b, for the prompt **3. Can the default session value be used for substitution when the request does not contain this session?**, select either **YES** or **NO**:

- – Select **YES** to cache one version of the object.

- – Select **NO** to cache two different versions of the object. OracleAS Web Cache serves one version to those requests that support the cookie or parameter and serves the other version to those requests that do not support the cookie or parameter.

6. In the Add Session Caching Policy dialog box, click **Submit**.

7. Associate the rule with URLs:

   a. In the Session Caching Policy Association page, select the newly-created rule, and then click **Change Rule Association.**

   The Change Session Caching Association dialog box appears.

   b. Select a selector from the right list, and then click **Make Association.**

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

   If the selector you require does not exist, create a caching rule for the pages the support session or personalized attributed cookie or embedded URL parameter, as described in "Configuring Caching Rules and Rule Association" on page 12-7. In Step 13 of the procedure, select a policy in the **Session Caching Policies** row of the Edit/Add Caching, Personalization, and Compression Rule dialog box.

8. Repeat Steps 2 through 7 for each policy.

9. Apply changes and restart OracleAS Web Cache:

   a. In the OracleAS Web Cache Manager main window, click **Apply Changes.**

   b. In the Cache Operations page, select a cache, and then click **Restart** to restart OracleAS Web Cache.

Configuring Support 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 OracleAS Web Cache to substitute sessions within HTML hyperlink tags with the session values obtained from a session cookie, embedded URL parameter, or `POST` body parameter.

> **See Also:** "Substituting Session Information in Session-Encoded URLs" on page 2-11 for an overview and an example scenario

To cache instructions for substituting session information in session-encoded URLs in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sessions**.

> **See Also:** "Configuring Support for Session-Encoded URLs" in Enterprise Manager Online Help for instructions

To cache instructions for substituting session information in session-encoded URLs in OracleAS Web Cache Manager:

1. As necessary, create session definitions for those pages with session-encoded URLs:

   a. In the navigator frame, select Rules for **Caching, Personalization, and Compression** > **Session Definitions**.

   The Session Definitions page appears.

   b. From the **For Site** list, select the Web site for which to create site-specific session definitions.

   c. Click **Add** or **Create.**

   The Edit/Add Session Definition dialog box appears.

   d. Select either **This Site Only** or **For All Sites.**

   e. In the **Session Name** 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. In the **Extract Value From** field, enter the cookie name in the **Cookie Name** field, or enter the embedded URL or POST body parameter in the **URL or POST body parameter** field.

   If you enter both a cookie name and 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 for the value of the embedded URL parameter.

   OracleAS Web Cache uses the default string for those requests without the value for an embedded URL parameter. For these requests, OracleAS Web Cache substitutes the value with a default string. The string defaults to default. For example, the following <A HREF=...> contains a session_ID parameter without a value:

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

   If the string is set to default, OracleAS Web Cache substitutes the value with default.

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

   If you want to instead require that the request get the cookie or parameter settings from the origin server, perform Step 3.

   h. In the **Comment** field, enter a description for the definition.

   i. Click **Submit.**

2. Create a caching rule for the objects that use the session, as described in "Configuring Caching Rules and Rule Association" on page 12-7.

In Step 13 of the procedure, select **Yes** in the **Session-Encoded URL** field of the Edit/Add Caching, Personalization, and Compression Rule dialog box to substitute session information in session-encoded URLs.

3. If you want to require that the request obtain the cookie, embedded URL parameter, or `POST` body parameter value from the origin server, perform these additional steps:

   a. Create a session caching policy for the page to track the session, as described in "Configuring Session or Personalized Attribute Caching Policies" on page 12-19.

   b. In Step 5a of the procedure, select **YES** as the response.

   c. In Step 5b of the procedure, select **NO** as the response.

4. Apply changes and restart OracleAS Web Cache:

   a. In the OracleAS Web Cache Manager main window, click **Apply Changes.**

   b. In the Cache Operations page, select a cache, and then click **Restart** to restart OracleAS Web Cache.

Configuring Support 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 OracleAS Web Cache to substitute personalized attributes within `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags with the values obtained from a cookie, embedded URL parameter, or `POST` parameter.

> **See Also:** "Personalized Attributes" on page 2-7 for an overview and an example scenario

To cache instructions for substituting session information in session-encoded URLs in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sessions**.

> **See Also:** "Configuring Support for Personalized Attributes" in Enterprise Manager Online Help for instructions

To create rules for personalized pages in OracleAS Web Cache Manager:

1. As necessary, create attribute definitions for those pages with personalized attributes:

   a. In the navigator frame, select **Rules for Caching, Personalization, and Compression** > **Session Definitions**.

   The Session Definitions page appears.

   b. From the **For Site** list, select the Web site for which to create site-specific personalized attribute definitions.

   c. Click **Add** or **Create.**

   The Edit/Add Session Definition dialog box appears.

   d. Select either **This Site Only** or **For All Sites.**

e. In the **Session Name** 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`.

f. In the **Extract Value From** field, enter the cookie name in the **Cookie Name** field, or enter the embedded URL or `POST` body parameter in the **URL or POST body parameter** field.

If you enter both a cookie name and 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 or POST body parameter** field.

---

g. In the **Default Value (Optional)** field, enter a default string for the value of the cookie, embedded URL parameter, or `POST` body parameter.

OracleAS Web Cache uses the default string for those requests without the value for the personalized attribute for pages containing the `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags. For these requests, OracleAS Web Cache uses the value of the default string. The string defaults to `default`. For example:

```
<B><!-- WEBCACHETAG="first_name01"-->default<!-- WEBCACHEEND--></B>
```

If you want to instead require that the request get the cookie or parameter settings from the origin server, perform Step 3.

h. In the **Comment** field, enter a description for the definition.

i. Click **Submit.**

2. Create a caching rule for the personalized page:

- Create a caching rule, as described in "Configuring Caching Rules and Rule Association" on page 12-7. In Step 13 of the procedure, select **Yes** in the **Session-Encoded URL** row of the Edit/Add Caching, Personalization, and Compression Rule dialog box to substitute information in personalized attributes.

- Configure the page with a `Surrogate-Control` response-header field, as described in "Surrogate-Control Response-Header Field" on page 12-30:

```
Surrogate-Control: content="webcache/1.0"
```

3. If you want to require that the request obtain the cookie, embedded URL parameter, or `POST` body parameter value from the origin server, perform these additional steps:

a. Create a caching policy for the page to track the session, as described in "Configuring Session or Personalized Attribute Caching Policies" on page 12-19.

b. In Step 5a of the procedure, select **YES** as the response.

c. In Step 5b of the procedure, select **NO** as the response.

4. Apply changes and restart OracleAS Web Cache:

a. In the OracleAS Web Cache Manager main window, click **Apply Changes.**

b. In the Cache Operations page, select a cache, and then click **Restart** to restart OracleAS Web Cache.

5. 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 12-34.

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 12-46

Example: Personalized Page Configuration

To understand how to cache personalized content, consider the HTML page `monthly.htm` in Figure 12–3 on page 12-26.

Figure 12–3   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.

To cache `monthly.htm` and its personalized content, take the following steps:

1. Map a personalized attribute of `TestMonth` to the embedded URL parameter `Month` in the Edit/Add Session Definition dialog box, as shown in the following figure:

2. In the Add Session Caching Policy dialog box, create a session caching policy that uses the session TestMonth, which uses the embedded URL parameter `Month`, as shown in the following figure:

> **See Also:** "Configuring Session or Personalized Attribute Caching Policies" on page 12-19 for more information about creating personalized attribute caching rules

3. Add the `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` HTML tags to `monthly.htm`.

```
Current Month is:
<!-- WEBCACHETAG="TestMonth"-->October<!-- WEBCACHEEND-->
```

4. Create a caching policy for `monthly.htm` in the Edit/Add Caching, Personalization, and Compression Rule dialog box:

a. In the **Session Caching Policies** row, choose the caching policy for the embedded URL `Month`.

b. In the **Session-Encoded URL** row, select **Yes** to cache substitution instructions for personalized attributes. The following figure shows the dialog box:

5. Apply the configuration changes:

   a. In the OracleAS Web Cache Manager main window, click **Apply Changes.**

   b. In the Cache Operations page, restart OracleAS Web Cache.

To verify that OracleAS Web Cache is caching `monthly.htm`:

1. Request an initial request for `monthly.htm` at the URL
   `monthly.htm?Month=October`. Because the initial request was forwarded by
   OracleAS Web Cache to the application Web server, the value `October` is set for
   the `Month` parameter. This initial request is inserted `monthly.htm` into the cache.

2. Send a subsequent request for `monthly.htm` to the URL
   `monthly.htm?Month=January`.

   OracleAS Web Cache substitutes `October` with the value of `January`, as shown
   in Figure 12–4 on page 12-29.

Figure 12–4   monthly.htm When Cached

Configuring 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, 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 and Rule Association" on page 12-7.

    In Step 13 of the procedure, select a session caching policy with a value of **Cache with session** and **Cache without session** in the **Session Caching Policies** row of the Edit/Add Caching, Personalization, and Compression 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 clients receive the response and session establishment from the application Web server. Subsequent redirected requests to the entry page carry the session, enabling the entry page to be served out of the cache.

Another solution 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 and Rule Association" on page 12-7.

> **Note:** Using the JavaScript solution, it is not necessary to create a session/personalized attribute related caching rule for the pages requiring sessions.

Using the Surrogate-Control Response Header as an Alternative to Caching Rules

In addition to, or as an alternative to, creating caching rules with Application Server Control Console or OracleAS Web Cache Manager, application developers can choose to store 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 Application Server Control Console or OracleAS Web Cache Manager interface, as well as allow other third-party caches to use OracleAS Web Cache caching attributes. All except the following attributes described in "Configuring Caching Rules and Rule Association" on page 12-7 are supported:

- ESI Output Permission

- 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 described in the following section.

> **See Also:** "About Cache Population" on page 2-1 for a description of how OracleAS Web Cache uses caching attributes from the `Surrogate-Control` response header and OracleAS Web Cache Manager to determine cache population

Surrogate-Control Response-Header Field

The `Surrogate-Control` response-header field enables application developers to specify caching attributes of an object. This response-header field enables the application Web server to override the caching rules configured through administrative interfaces Oracle Enterprise Manager 10*g* Application Server Control Console or OracleAS Web Cache Manager.

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

```
Surrogate-Control:[content=content_type, content_type,..]
[no-store][max-age=expiration_time[+removal_time]]
[vary=headers(header header...)][coookie(cookie_name cookie_name...)]
[compress=yes|no]
```

Table 12–7 describes the supported control directives.

Table 12–7   Control Directives for Surrogate-Control

Control Directive	Description
content	Specify what kind of processing is required: ■ `"ORAESI/9.0.4"` to process ESI tags with Oracle-proprietary additions for content assembly and partial page caching. `"ORAESI/9.0.4"` supports all the ESI tags provided by OracleAS Web Cache in 10*g* (9.0.4) and later releases. ■ `"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 OracleAS Web Cache in Release 2 (9.0.2 and 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 ■ `"ESI-INV/1.0"` to process `<esi:invalidate>` tags ■ `"webcache/1.0"` to process the `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags for personalized attributes `"ORAESI/9.0.2"`, `"ESI/1.0"`, and `"ESI-Inline/1.0"` are subsets of `"ORAESI/9.0.4"`. In this release, you specify only `"ORAESI/9.0.4"` for ESI assembly, `"ESI-INV/1.0"` for inline invalidation, or `"webcache/1.0"` for personalized attributes. **See Also:** Table 16–2 on page 16-7 for further information about the ESI tags supported for each processing version
no-store	Specify that OracleAS Web Cache not cache the object.
vary	Specify the HTTP request headers or cookies from which OracleAS Web Cache will use to cache and identify multiple-version objects. Use the following format: `vary=[headers(header[/f] *);] [cookies(cookie_name[/f] *)]` Specify /f to instruct OracleAS Web Cache to only cache versions of the object based on the existence of the HTTP request headers or cookies. Exclude /f to instruct OracleAS Web Cache to cache versions of the object, regardless of whether the HTTP request headers or cookies exist. Usage notes: ■ Specify at least one HTTP header or cookie. ■ If you specify both HTTP request headers and cookies, specify the HTTP request header before the cookies. ■ Use zero or more spaces between the parentheses and semicolons. ■ When specifying multiple HTTP request headers or cookies, use one or more spaces between the HTTP request headers and cookie names.

Table 12–7   (Cont.) Control Directives for Surrogate-Control

Control Directive	Description
compress	Specify `yes` for OracleAS Web Cache to serve compressed cacheable and non-cacheable objects to all browser types; specify `no` for OracleAS Web Cache to not serve compressed cacheable and non-cacheable objects to browsers.
	This control directive does not enable you to specify browser types. If you specify yes and need to limit the browser types, specify a compression caching rule in OracleAS Web Cache Manager, as described in Step 11 of "Task 1: Create Caching Rules" on page 12-7.
	Notes: Even if you enable compression, OracleAS Web Cache does not compress the following:
	■ GIF, JPEG, PDF, PNG, SWF, and ZIP files
	■ Responses containing a `Content-Encoding` response-header field, which is typically used to denote compression
	■ Responses containing a `Content-Disposition` response-header field, which is typically used for attachments
	(Responses with `Content-Disposition` response-header fields show incorrect file names when they are compressed.)
	■ For certain browsers, JavaScript files
	Compressed JavaScript files cause some browsers to behave erratically and possibly fail. This issue only affects files that are referenced with the `src` attribute of the `script` tag; it does not include files that contain inline JavaScript.
	OracleAS Web Cache does not compress JavaScript files for Netscape 4.*x* and Internet Explorer 5.5 browsers. For other browsers, it compresses the file if compression is enabled.
	■ For certain browsers, cascading style sheets (`.css`)
	OracleAS Web Cache does not compress cascading style sheets for Netscape 4.*x* and Internet Explorer 5.5 browsers. With these browsers, compressed cascading style sheets can cause background attributes, such as background images, to not appear in the output.
max-age	Specify that OracleAS Web Cache cache the object.
	Specify the time, in seconds, to expire the object after it enters the cache. Optionally, specify the time, in seconds, to remove the object 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 object never expires

Usage Notes

■   Control directives are case sensitive.

■   `content="ORAESI/9.0.4"`,`content="ESI-Inline/1.0"`, `content="ESI-INV/1.0"`,`content="ESI/1.0"` are mutually exclusive with `content="webcache/1.0"`

> **See Also:** http://www.esi.org/spec.html for the *Edge Architecture Specification*, which contains specification information about the `Surrogate-Control` response header

Example Usage

In the following example, the `Surrogate-Control` response-header field specifies that the object is to expire 30 seconds after it enters the cache and be removed 60

seconds after expiration. It also specifies that the object contains ESI tags that require processing:

```
Surrogate-Control: max-age=30+60, content="ORAESI/9.0.4"
```

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

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

In the following example, the `Surrogate-Control` response-header field specifies ESI processing with the `content` control directive. The `vary` control directive specifies to cache versions of the multiple-version object, regardless of whether the request contains the HTTP `Accept` request header.

```
Surrogate-Control: content="ORAESI/9.0.4", vary=headers(Accept)
```

In the following similar example, the `Surrogate-Control` response-header field specifies ESI processing with the `content` control directive. The `vary` control directive specifies to cache versions of the multiple-version object only if the request contains the `Accept` and `MyCustomHeader` headers and `news` and `sports` cookies.

```
Surrogate-Control: content="ORAESI/9.0.4", vary=headers(Accept/f);cookies(news/f
sports/f)
```

Configuring Rules for Content Assembly and Partial Page Caching

> **See Also:** "About Edge Side Includes (ESI) for Partial Page Caching" on page 2-13 for an overview of partial page caching

This section describes how to enable dynamic assembly of Web pages with fragments and how to 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 12-34.

   b. In the template page, configure the HTTP response with the `Surrogate-Control` response-header field. For example:

   ```
   Surrogate-Control: max-age=30+60, content="ORAESI/9.0.4"
   ```

    c. If the `Surrogate-Control` response-header field does not include all the caching attributes required for the template page, create a caching rule for the page.

2. Configure the fetchable fragments:

- 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, create a caching rule for the fragment.

    See Also:

- Chapter 16, "Edge Side Includes (ESI) Language Tags" for further information about ESI markup tags

- "Using the Surrogate-Control Response Header as an Alternative to Caching Rules" on page 12-30 for further information about configuring the `Surrogate-Control` response-header field

- "Configuring Caching Rules and Rule Association" on page 12-7 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 recommends using ESI for simple personalization when you are utilizing other ESI features, otherwise continue to use the methods described in "Configuring Rules for Popular Pages with Session Establishment" on page 12-29.

For example, the following HTML excerpt uses the `<!-- WEBCACHETAG-->` and`<!-- WEBCACHEEND-->` tags to substitute a user's name based on the value the client 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 16-5

- "ESI vars Tag" on page 16-32

Examples of ESI Usage

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

- Example of a Portal Site Implementation

- Example of Simple Personalization with Variable Expressions

Example of a Portal Site Implementation

Figure 12–5 shows a portal site response page,
`http://www.company.com/servlet/oportal?username=Mark`, for a registered
user named Mark.

Figure 12–5   Portal Site Page

This page is assembled by OracleAS Web Cache. A template page configured with ESI
markup tags for a personalized greeting, weather, stocks, promotional advertisement,
news, and sports fragments is assembled based on Mark's preferences. For example,
because 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, OracleAS 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 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 16-23 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.

Example 12–1 shows `portal.esi` with `<esi:inline>` tags.

Example 12–1 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: 50F
    </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 84.99
    </esi:inline>
```

```
      <BR>
      <esi:inline name="/ticker_ORCL">
        ORCL 13.379
      </esi:inline>
      <BR>
      <esi:inline name="/ticker_YHOO">
        YHOO 27.15
      </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>
     Tech Spending Growth Indexes Little Changes
     Home Sales Hit Record High
     Gas Prices Dip Again
  </TR>
</TABLE>
</esi:inline>

<esi:inline name="/Sports_News_Soccer" >
<!-- Personalized Sports News -->
Latest Sports News for Soccer

<TABLE>
  <TR>
  Preparation for World Cup
  Youth Cup game on a Sunday
  Latest Scores
  </TR>
</TABLE>
</esi:inline>
```

```
<esi:inline name="/Common_Fragment_5" >
...
</esi:inline>
```

Example 12–2 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.

Example 12–2   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>
```

Example 12–3 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.

Example 12–3   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: 50F
    </TD>
  </TR>
</TABLE>
</esi:inline>
```

Example 12–4 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 markup separates the unique information from the shared information, maximizing the reduction of cache updates and space consumption of personal stock selection.

Example 12–4   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 84.99
    </esi:inline>
    <BR>
    <esi:inline name="/ticker_ORCL">
      ORCL 13.379
    </esi:inline>
    <BR>
    <esi:inline name="/ticker_YHOO">
      YHOO 27.15
    </esi:inline>
    <TD>
  </TR>
</TABLE>
</esi:inline>
```

Example 12–5 shows the markup for referencing an advertisement in the Promotion section. `promotionID` is the based on the user's identification.

Example 12–5   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 12-40 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. Example 12–6 shows the markup.

Example 12–6   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>
      Tech Spending Growth Indexes Little Changes
```

```
      Home Sales Hit Record High
      Gas Prices Dip Again
   </TR>
</TABLE>
</esi:inline>


<esi:inline name="/Sports_News_Soccer" >
<!-- Personalized Sports News -->
Latest Sports News for Soccer
<TABLE>
   <TR>
   Preparation for World Cup
   Youth Cup game on a Sunday
   Latest Scores
   </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.

Example 12–7 shows portal.esi with <esi:include> tags.

Example 12–7   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>
```

Example 12–8 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:

Example 12–8   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"/>
```

Example 12–9 shows `GetProfile`, which provides access to the `city`, `state`, `news`, and `sports` environment variables.

Example 12–9   portal.esi Example: GetProfile File with Environment Variables

```
<?xml version="1.0"?>
<esi-environment esiversion="ORAESI/9.0.4">
  <city>San_Francisco</city>
  <state>CA</state>
  <news>finance</news>
  <sports>soccer</sports>
</esi-environment>
```

Example 12–10 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. The parameter `username` is the registered user's name. This markup enables the personalized greeting to be included in the cacheable template page.

Example 12–10   portal.esi Example with vars tag: Personalized Greeting

```
<esi:vars>Welcome, $(QUERY_STRING{username})!</esi:vars>
```

Example 12–11 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.

Example 12–11   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 Example 12–12. 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.

Example 12–12   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 Example 12–13. It includes fragments for three stock quotes: IBM, ORCL, and YHOO.

Example 12–13   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>
    </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 Example 12–14. 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.

Example 12–14   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>
```

Example 12–15 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, OracleAS Web Cache is able to cache the template and integrate the dynamic advertisement into the template.

Example 12–15   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 Example 12–16, the response to the included image fragment for the banner is not cacheable. When a user requests this page, OracleAS 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.

Example 12–16   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 Example 12–17, the next time the user reloads the page, `Advert` generates another banner for the request.

Example 12–17   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`, a link to `www.oracle.com` appears in the banner's place.

Example 12–18 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`.

Example 12–18   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 5 of , 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 Example 12–19. Its HTML code uses PL/SQL for an HTML form with a text box in it.

Example 12–19 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>');
```

Example 12–20 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.

Example 12–20 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>');
```

13

Sending Invalidation Requests

This chapter explains how to send invalidation requests to OracleAS Web Cache.

This chapter contains these topics:

- Invalidation from External Sources
- Inline Invalidation in HTTP Responses
- Reducing Invalidation Overhead
- Using Search Keys in Surrogate-Key Response Header and Invalidation Requests

Invalidation from External Sources

This section provides the following topics for understanding how to send invalidation requests from external sources:

- Role of the invalidator and administrator Accounts
- Format of Invalidation Requests
- Methods for Sending Requests

Role of the invalidator and administrator Accounts

To invalidate objects in the cache, send an HTTP `POST` request from the `invalidator`, `ias_admin`, or `administrator` account through an invalidation listening port.

The `invalidator` account is an administrator authorized to send invalidation requests. In a **cache hierarchy** of OracleAS Web Cache servers, the **central cache** or **provider cache** uses the `invalidator` account name and password of the remote or subscriber OracleAS Web Cache server. The invalidation request specifies the objects to invalidate, as well as the site host name of the objects. The site host name is compared with the IP address of the cache from which the invalidation request was propagated. If there is a match, the cache processes the invalidation request. Otherwise, the request is rejected.

While the `ias_admin` or `administrator` account typically sends configuration and administration requests to OracleAS Web Cache, you can also use it to send invalidation requests. Because the `ias_admin` or `administrator` account is authorized with more privileges than the `invalidator` account, an invalidation request sent by an `ias_admin` or `administrator` account does not need to specify the site host name. Invalidation requests sent from Application Server Control Console use the `administrator` account; invalidation requests send from OracleAS Web Cache Manager use the `administrator` account.

For automatic propagation of invalidation messages, OracleAS Web Cache passes the encoded `invalidator` password in the page request between the central and **remote cache** or the provider and **subscriber cache** during the hierarchy registration process. This HTTP traffic is susceptible to network sniffing. If the network is unprotected and insecure, configure HTTPS ports as follows:

- From the Ports page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**) or the Listen Ports page of OracleAS Web Cache Manager (**Ports** > **Listen Ports**) of the central or provider cache, disable the default HTTP port and configure an HTTPS port in its place. See "Task 6: Configure OracleAS Web Cache with Listening Ports for Client Requests" on page 8-19.

- From the Ports page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**) or the Operations Ports page of OracleAS Web Cache Manager (**Ports** > **Operations Ports**) of the remote or subscriber cache, disable the default HTTP port for invalidation and configure an HTTPS port in its place. See "Task 8: Configure OracleAS Web Cache with Operations Ports" on page 8-22.

> **See Also:**
>
> - Step 1 in "Task 2: Modify Security Settings" on page 8-7 for further information about setting passwords
>
> - "Invalidation" on page 2-2 for further information about invalidation propagation
>
> - Section 7.3.3, "Portal Web Cache Settings" in *Oracle Application Server Portal Configuration Guide* for information about additional settings required for OracleAS Portal configurations

Format of Invalidation Requests

The HTTP `POST` requests are written in **Extensible Markup Language (XML)** syntax. The contents of the XML request body instructs the cache which URLs to mark as invalid.

Propagation of Invalidation Messages

Propagation of invalidation messages from one OracleAS Web Cache server to another occurs in the following deployments:

- Cache hierarchy whereby one OracleAS Web Cache server acts as an origin server to another OracleAS Web Cache server

- Cache cluster with multiple OracleAS Web Cache servers

Invalidation in Hierarchies

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

Figure 13–1 on page 13-3 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 stores content from an application Web server, the remote cache stores 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 13–1   Scenario 1: Invalidating Content in a Distributed Cache Hierarchy

Figure 13–2 on page 13-4 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 13–2   Scenario 2: Invalidating Content in an ESI Cache Hierarchy

When an invalidation message is sent to a central or provider cache to refresh content, the central or provider cache automatically propagates the invalidation message to the remote or subscriber cache to ensure consistency.

To ensure that the central or provider cache only invalidates its content, the remote or 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 was propagated. If there is a match, the remote or subscriber cache processes the invalidation request. Otherwise, the request is rejected. For distributed cache hierarchies, the site host name for the central and remote caches should be configured to be identical, making a mismatch unlikely.

> **See Also:**
>
> - "Cache Hierarchies" on page 1-10 for an overview of cache hierarchies
>
> - Chapter 11 for instructions on configuring a cache hierarchy
>
> - "About Edge Side Includes (ESI) for Partial Page Caching" on page 2-13 for an overview of ESI
>
> - Chapter 13 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 OracleAS Web Cache propagates invalidation messages, the cache that received the invalidation request acts as the **invalidation coordinator** for that request. The coordinator propagates the invalidation messages to the other cluster members. The coordinator waits for responses from all cluster members. When the propagation completes, the coordinator returns a message to the sender that lists, for each cluster member, the cluster member name, the status of the invalidation request, and the number of objects invalidated.

If any cluster member cannot be reached, OracleAS Web Cache returns an error message and does not propagate the invalidation messages.

During a cache cluster upgrade, you upgrade one cache cluster member at a time. The caches will continue to respond to requests. However, because other cluster members have a different version of the configuration, the caches will not forward invalidation messages to those cache cluster members operating with a different version. Instead, if the requested object is not cached by that cache or by cluster members with the same version of the configuration, OracleAS Web Cache forwards the request to the origin server.

When the cache cluster members are not running the same version of OracleAS Web Cache, you can still invalidate objects, and you can propagate the invalidation to other cluster members, but the invalidation message must originate from the cache that is operating with the earlier version of OracleAS Web Cache.

> **See Also:**
>
> - *Oracle Application Server Upgrade and Compatibility Guide* for more information about upgrading OracleAS Web Cache to 10*g* Release 2 (10.1.2), including information about upgrading cache cluster members
>
> - Chapter 13 for more information about invalidation

Methods for Sending Requests

As shown in Figure 13–3 on page 13-6, you send invalidation requests using one of the following methods:

- Manually, using Oracle Enterprise Manager 10*g* Application Server Control Console, OracleAS Web Cache Manager, or `telnet`

- Automatically, using database triggers, scripts, or application logic

Figure 13–3 Invalidation

Specifically, you can use one of the following methods:

- Telnet to Send Invalidation Requests

- Oracle Enterprise Manager 10g Application Server Control Console or OracleAS Web Cache Manager to Send Invalidation Requests

- Application Program Interfaces (APIs) for Automated Invalidation Requests

- Database Triggers for Automated Invalidation Requests

- Scripts for Automated Invalidations

Telnet to Send Invalidation Requests

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

For example, if you are using telnet, you send an invalidation request using the following procedure:

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

   ```
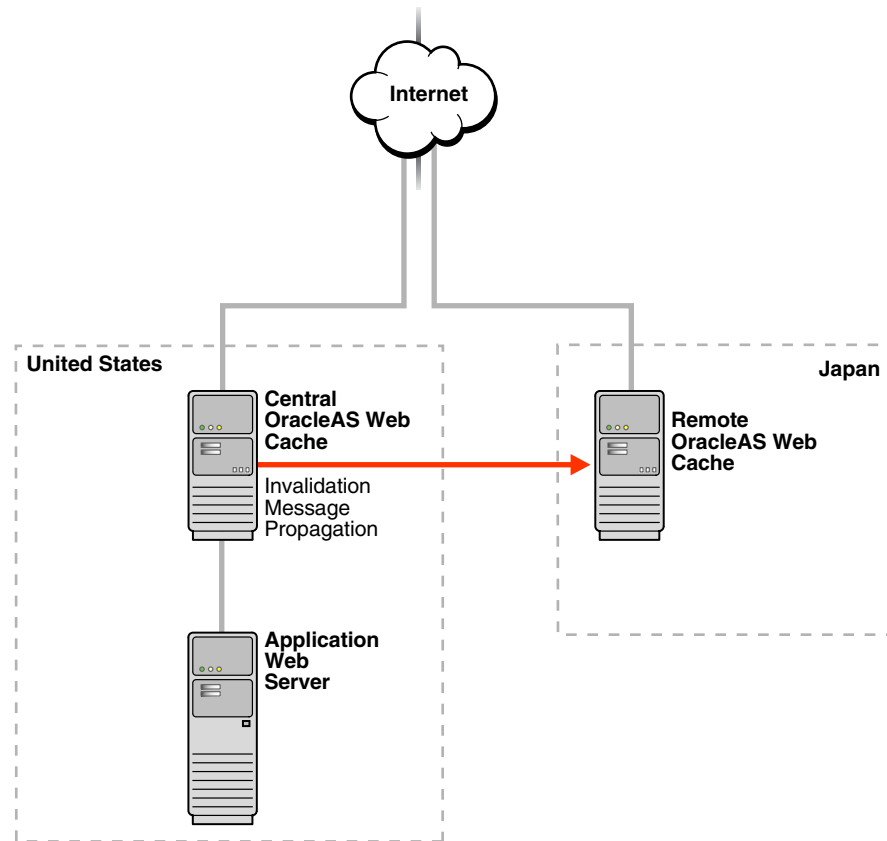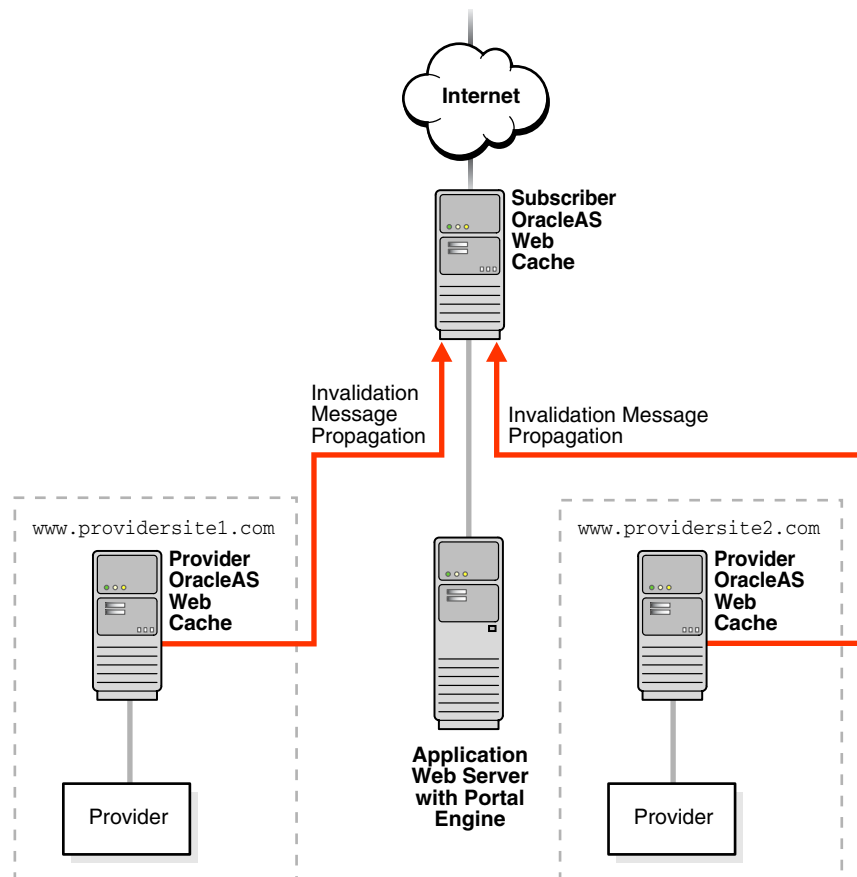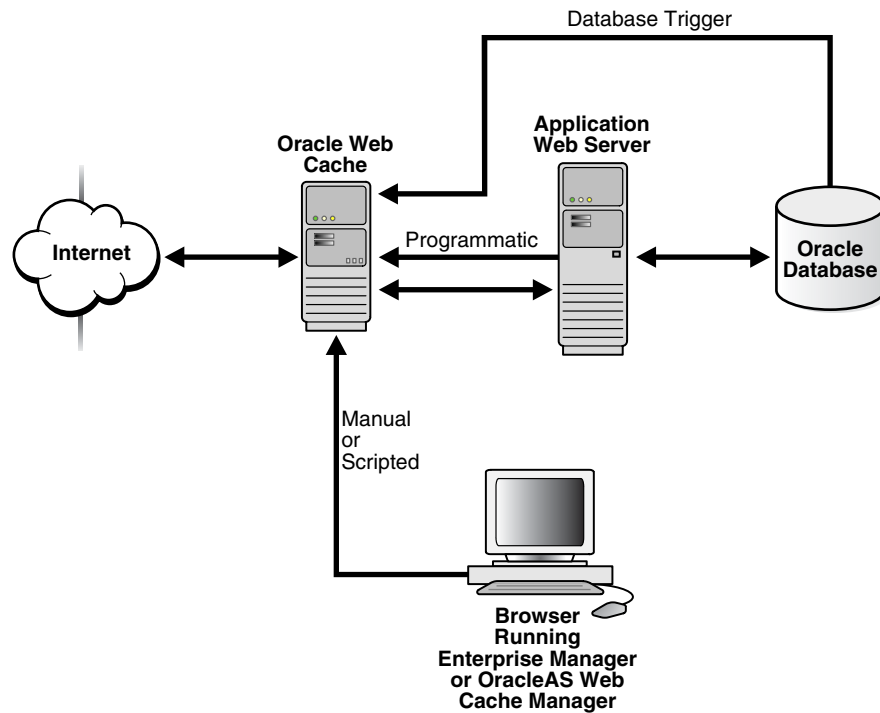   telnet web_cache_host invalidation_port
   ```

2. Specify a POST message header and authenticate the invalidator account 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
   ```

   The following shows an example of the Authorization line:

   ```
   Authorization: BASIC aW52YWxpZGF0b3I6aW52YWxpZGF0b3I=
   ```

In this example, `aW52YWxpZGF0b3I6aW52YWxpZGF0b3I=` is the invalidator user name and password (`invalidator:invalidator`) encoded.

> **See Also:**
>
> - http://www.rfc-editor.org/ for information about password Base64 encoding
>
> - `readme.examples.html` in the `$ORACLE_HOME/webcache/docs` directory on UNIX and *ORACLE_HOME*`\webcache\docs` 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 6-11 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 Response Syntax

- Invalidation Preview Request Syntax

- Invalidation Preview Response Syntax

Invalidation Request Syntax

Use the following syntax to invalidate objects 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 objects 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|SEARCHKEY"
                 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 that 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.

Note the following:

- 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/90400/WCSinvalidation.dtd`"

The root element `INVALIDATION` contains one or more of the attributes and elements described in Table 13–1 on page 13-9.

Table 13–1   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.*x* and later, always use `VERSION="WCS-1.1"`, unless you require previous 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 requires the `SYSTEMINFO` element.
`SYSTEMINFO` element	Required element in the `SYSTEM` element
	The possible `NAME/VALUE` pairs are:
	■ `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`.
	■ `NAME="WCS_DISCONNECTED_MODE_OK" VALUE="TRUE|FALSE"`
	This pair specifies how soon invalidation takes place. If `WCS_DISCONNECTED_MODE_OK` is `TRUE`, invalidation is not immediately performed. The invalidation response is sent as soon as the invalidation request is received. Set this element to `TRUE`, if you do not want to wait for the invalidation result. If `WCS_DISCONNECTED_MODE_OK` is `FALSE`, invalidation is completed immediately and the invalidation result is sent.
	The default is `FALSE`.
`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 objects 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` account is sending the request.
`ADVANCEDSELECTOR` element	`URIPREFIX` attribute: Required attribute of the `ADVANCEDSELECTOR` element. Specify the path prefix of the objects to be invalidated. The path prefix 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` account is sending the request.
	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 (\).

Table 13–1   (Cont.) INVALIDATION Elements and Attributes

Invalidation Element/ Attribute	Description
	URIEXP attribute: Optional attribute of the ADVANCEDSELECTOR element. Specify the URL of the objects to be invalidated underneath the URIPREFIX. If no value is entered, 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 client browsers send to OracleAS Web Cache and the URL that OracleAS Web Cache uses internally for that request are different. When OracleAS 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, OracleAS 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.
	HOST attribute: This attribute is required if the URIPREFIX value does not include *host_name:port* and the invalidator account 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.
	METHOD attribute: Optional attribute of the ADVANCEDSELECTOR element. Specify **HTTP request method** (GET or POST) of the objects to be invalidated. GET is the default value.
	BODYEXP attribute: Optional attribute of the ADVANCEDSELECTOR element. If the METHOD is POST, specify HTTP POST body of the objects to be invalidated.
	Note: When the invalidation request is sent, OracleAS 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. Use the following attributes:
	■ NAME attribute: Required attribute for the COOKIE element attribute. Specify the cookie name to invalidate multiple-version objects based on the cookie.
	■ VALUE attribute: Optional attribute for the COOKIE element. Specify the value of the cookie. If no value is present, only objects with the named cookie but without a value are invalidated.
	If you specify a cookie that was mistakenly specified for both a multiple-version object and a session caching policy, invalidation is based on any occurrence of the cookie. To avoid excessive invalidation, configure distinct cookies for multiple-version objects and session caching policies.
	See Also:
	■ "Task 1: Create Caching Rules" on page 12-7 to create caching rules for multiple-version objects
	■ "Configuring Cookie Definitions for Multiple-Version Objects Containing Cookies" on page 12-15 to specify cookie definitions
	■ "Configuring Session or Personalized Attribute Caching Policies" on page 12-19 to specify session caching policies

Table 13–1   (Cont.) INVALIDATION Elements and Attributes

Invalidation Element/ Attribute	Description
HEADER element	Optional element in the invalidation request. Use the following attributes: ■　NAME attribute: Required attribute for the HEADER element. Specify the **HTTP request header** and its value to invalidate based multiple-version objects based on the request header. ■　VALUE attribute: Optional attribute for the HEADER element. Specify the value of the header. **See Also:** "Task 1: Create Caching Rules" on page 12-7 to create caching rules for multiple-version objects
OTHER element	Optional element in the invalidation request. Use the following attributes: ■　NAME attribute: Required attribute of the OTHER element. NAME supports the following values: - URI to specify a match of the URL underneath the URIPREFIX - BODY to specify a match of the HTTP POST body - QUERYSTRING_PARAMETER to specify a match of an embedded URL parameter - SEARCHKEY to specify a match of a search key in the Surrogate-Key response header ■　TYPE attribute: Required attribute for URI, BODY, and QUERYSTRING_ PARAMETER. This attribute is not recognized for SEARCHKEY. TYPE supports the following values: - SUBSTRING to specify an exact string match for QUERYSTRING_PARAMETER and a substring match for URI and BODY. - REGEX to specify a regular expression match ■　VALUE attribute: Required attribute for URI, BODY, QUERYSTRING_PARAMETER, and SEARCHKEY. Specify the value of URI, BODY, QUERYSTRING_PARAMETER, or SEARCHKEY. If you specify a TYPE of REGEX, then escape regular expression characters that you want to interpreted literally with a backslash (\). **See Also:** ■　"Invalidation Examples" on page 13-23 to optimize advanced invalidations ■　"Using Search Keys in Surrogate-Key Response Header and Invalidation Requests" on page 13-39 to configure search keys
ACTION element	Required element in the invalidation request REMOVALTTL attribute Optional attribute of the ACTION element. Specify the maximum time that objects 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 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: `[13/Jul/2005:19:26:46 +0000] [notification 11748] [invalidation] [ecid: 21085932167,0] Invalidation with INFO 'INFO_comment' has returned with status 'status'; number of objects invalidated: 'number'.`

> **Note:** The following special XML characters must be escaped in the fields: ampersand (&) with "&", greater than sign (>) with "&gt", less than sign (<) with "&lt", double quotes (") with "&quot", and single quotes (') with "'".

> **Note:** OracleAS 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-1 for further information about invalidation request syntax

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-1.1">
   <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-1.1">
   <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|SEARCHKEY"
                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 13–2. `BASICSELECTOR` and `ADVANCEDSELECTOR` are described in Table 13–1 on page 13-9.

Table 13–2   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. The `SYSTEM` element requires the `SYSTEMINFO` element.

Table 13–2   (Cont.) INVALIDATIONRESULT Elements and Attributes

Invalidation Element/ Attribute	Description
SYSTEMINFO element	Required element in the SYSTEM element. The possible NAME/VALUE pairs are as follows: ■ NAME="WCS_CACHE_NAME" VALUE="*string*" This pair specifies the name of the cache. ■ NAME="WCS_DISCONNECTED_MODE_OK" VALUE="TRUE|FALSE" This pair specifies how soon invalidation takes place. If WCS_DISCONNECTED_MODE_OK is TRUE, invalidation is not immediately performed. The invalidation response is sent as soon as the invalidation request is received. Set this element to TRUE, if you do not want to wait for the invalidation result. If WCS_DISCONNECTED_MODE_OK is FALSE, invalidation is completed immediately and the invalidation result is sent. The default is FALSE.
RESULT element	Use the following attributes: ■ ID attribute: Sequence number of all the invalidation objects sent in the invalidation response. If there are multiple selectors specified in the invalidation request, 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 objects that are not cacheable - URI NOT FOUND for objects not found ■ NUMINV attribute: Number of objects 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-1 for
> further information about invalidation response syntax

Invalidation Preview Request Syntax

To test invalidation, use the following syntax to preview the list of BASICSELECTOR objects 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"/>
</INVALIDATIONPREVIEW>
```

Use the following syntax to preview the list of ADVANCEDSELECTOR objects 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|SEARCHKEY"
                   TYPE="SUBSTRING|REGEX"
                   VALUE="value"/>
        </ADVANCEDSELECTOR>
</INVALIDATIONPREVIEW>
```

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 13–3. BASICSELECTOR and ADVANCEDSELECTOR are described in Table 13–1 on page 13-9.

Table 13–3   INVALIDATIONPREVIEW Attributes

Invalidation Element/ Attribute	Description
VERSION attribute	Required attribute in the invalidation preview
	Use 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 invalidation preview
	Enter the number representing the first object to be listed. OracleAS Web Cache begins the count of objects with the number 0.
MAXNUM attribute	Required attribute in the invalidation preview
	Enter the number of objects to be listed.
	If fewer objects than the number specified meet the invalidation criteria, OracleAS Web Cache lists only the URLs for those objects that meet the criteria.
	If more objects than the number specified meet the invalidation criteria, OracleAS Web Cache lists the URLs for the number of objects requested. It also returns the total number of objects that meet the invalidation criteria. To obtain the list of URLs for addition objects, send another preview request with a different STARTNUM that specifies the start of the next set of objects.

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 that the response is an invalidation preview response using the WCSinvalidation.dtd file as the XML document type.

Note the following:

- 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/90400/WCSinvalidation.dtd"

The root element INVALIDATIONPREVIEWRESULT contains one or more of the attributes and elements described in Table 13–4. BASICSELECTOR and ADVANCEDSELECTOR are described in Table 13–1 on page 13-9.

Table 13–4   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 objects that are not cacheable ■ URI NOT FOUND for objects not found
STARTNUM attribute	Number representing the first object 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

Oracle Enterprise Manager 10*g* Application Server Control Console or OracleAS Web Cache Manager to Send Invalidation Requests

Application Server Control Console or OracleAS 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 one of these interface 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.

Application Server Control Console provides a step-by-step wizard that guides you through the invalidation process. To get started, navigate to **Web Cache Home** page > **Administration** tab > **Operations** > **Invalidation**. Use the online help for the wizard pages for further assistance.

OracleAS Web Cache Manager enables you to send either basic invalidation request for invalidation one object, or an advanced invalidation request for multiple objects, as described in the following topics:

- Submitting Basic Invalidation Requests

- Submitting Advanced Invalidation Requests

> **Note:**   If you receive the following error when you submit
> invalidation requests from the Basic Content Invalidation or
> Advanced Content Invalidation pages, restart the `cache` or `admin`
> server processes.
>
> ```
> Internal error: can't connect to OracleAS Web Cache
> Invalidation Listening Port
> ```
>
> If you change the property of an invalidation port, restart the `cache`
> server process. If you change the password for the `administrator`
> account in the Security page, restart the `cache` and `admin` server
> process. Until your restart the `cache` server process for either
> configuration change, invalidation requests will return the error.
>
> See Chapter 7 for restart instructions.

Submitting Basic Invalidation Requests

To send a basic invalidation request using OracleAS Web Cache Manager:

1. In the navigator frame, select **Operations** > **Basic Content Invalidation.**

    The Basic Content Invalidation page appears in the right pane.

2. From the **For Cache** list, select a cache. (More than one cache is listed only if you
    configured a cache cluster. If you configured the cluster to propagate invalidation,
    the cache you select is designated the invalidation coordinator, which will
    propagate the invalidation request to other cache cluster members. If you did not
    configure the cluster to propagate invalidation, the cache you select is the only
    cache from which objects are invalidated.)

3. In the **Search Criteria** section, select the search criteria:

    ■ **Remove all cached objects:** Select to remove all objects from the cache.

    ■ **Enter exact URL for removal**: Specify the URL of the objects to be invalidated.
        Include the complete path and file name.

        Note: Because OracleAS Web Cache escapes the following characters, you can
        enter them in this field: ampersand (`&`), greater than sign (`>`), less than sign (`<`),
        double quotes (`"`), and single quotes (`'`).

4. Optionally, you can preview the list of objects to be invalidated to ensure that you
    are removing only the objects you want to remove. To preview the list of objects:

    a. In the **Action** section, choose **Preview list of objects that match invalidation
        criteria**.

    b. Specify the **Object Range:**

        – **From:** Enter the number representing the first object to be listed. OracleAS
           Web Cache begins the count of objects with the number 0.

        – **To:** Enter the number of objects to be listed.

           If fewer objects than the number specified meet the invalidation criteria,
           OracleAS Web Cache lists the URLs for only those objects that meet the
           criteria.

           If more objects than the number specified meet the invalidation criteria,
           OracleAS Web Cache lists the URLs for the number of objects requested. It
           also returns the total number of objects that meet the invalidation criteria.
           To obtain the list of URLs for addition objects, send another preview

request with a different **From** number that specifies the start of the next set of objects.

c.   Click **Submit.**

OracleAS Web Cache displays the Invalidation Preview Results message box, which lists the objects that meet the invalidation criteria. OracleAS Web Cache Manager lists only those objects that are valid. Although the cache may contain objects that are expired or that have been invalidated, those objects are not listed.

If the objects listed are those that you want to invalidate, continue with the next step. If they are not, modify the invalidation criteria and preview the list again.

5.   In the **Action** section, select one of the following options to specify how to process invalid objects:

   ■   **Remove immediately:** OracleAS Web Cache marks objects as invalid and removes them immediately. A object is refreshed from the origin server when the cache receives the next request for it.

   ■   **Remove objects no later than** *<number>* *<time>* **after submission:** OracleAS Web Cache marks objects as invalid and then refreshes them based on origin server capacity. Enter the maximum time in which the objects can reside in the cache.

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

---

6.   Click **Submit.**

OracleAS Web Cache processes the invalidation request, and returns the Cache Cleanup Result dialog box that the shows the invalidation status and the number of objects invalidated. The following figure shows the dialog box:

In a cache cluster environment, if **Propagate Invalidation** is enabled, OracleAS Web Cache sends the invalidation request to one cluster member who acts as the **invalidation coordinator**. The coordinator propagates the invalidation request to other cluster members. When the invalidation has been completed for all cluster members, OracleAS 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:**
>
> - "Propagating Configuration Changes to Cache Cluster Members" on page 10-10 for information about enabling invalidation propagation
>
> - "Invalidation in Cache Clusters" on page 13-5 for further information about invalidation propagation in a cache cluster

Submitting Advanced Invalidation Requests

To send an advanced invalidation request using OracleAS Web Cache Manager:

1. In the navigator frame, select **Operations** > **Advanced Content Invalidation**.

    The Advanced Content Invalidation page appears in the right pane.

2. From the **For Cache** list, select a cache. (More than one cache is listed only if you configured a cache cluster.)

3. In the **Search Criteria** section, select the search criteria:

    > **Note:** Because OracleAS Web Cache escapes the following characters, you can enter them in the search criteria fields: ampersand (&), greater than sign (>), less than sign (<), double quotes ("), and single quotes (').

    - **URL Path Prefix:** *Required.* Specify the path prefix of the objects to be invalidated. The path prefix 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 (`\`).

    - **Host Name:** *Optional.* Specify the host name and port number of the site (*host_name*`:`*port*). Port 80 is the default port for HTTP.

      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 objects to be invalidated. The default value is `GET`.

    - **URL Expression:** *Optional.* Specify the URL of the objects to be invalidated under the **URL Path Prefix.** Then, specify how to match by selecting either **substring** for an exact string match or **regular expression** for a regular expression match.

      If no value is entered, everything under the **URL Path Prefix** is matched.

    - **POST Body Expression:** *Optional.* If **POST** is selected for the **HTTP Method**, enter the HTTP `POST` body of the objects to be invalidated, and then specify how to match by selecting either **substring** for a substring string match or **regular expression** for a regular expression match.

> **Note:** If **regular expression** is selected for the **URL Expression** or **POST Body Expression** fields, OracleAS Web Cache interprets the following reserved regular expression characters: periods (`.`), question marks (`?`), asterisks (`*`), brackets (`[]`), curly braces (`{}`), carets (`^`), dollar signs (`$`), and backslashes (`\`). To interpret these characters literally, escape them with a backslash (`\`).

4. Optionally, in the **Cookie/Header Information** section, specify the use of cookie names or HTTP request headers used for multiple-version objects in the search criteria:

   a. From the list, select **Cookie** or **Header.**

   b. Provide the following information:

   – **Cookie:** Enter the cookie name used by multiple-version objects to be invalidated in the **Name** field, and enter its value in the **Value** field.

   > **Note:** If you specify a cookie that was mistakenly specified for both a multiple-version object and a session caching policy, invalidation is based on any occurrence of the cookie. To avoid excessive invalidation, configure distinct cookies for multiple-version objects (**Rules for Caching, Personalization, and Compression** > **Cookie Definitions**) and session caching policies (**Rules for Caching, Personalization, and Compression** > **Session Definitions**).

   – **Header:** Enter the HTTP request header used by the objects to be invalidated in the **Name** field, and enter its value in the **Value** field.

   > **See Also:** "Task 1: Create Caching Rules" on page 12-7 to create caching rules for multiple-version objects

5. Optionally, in the **URL Parameters** section, enter the name of the embedded URL parameter used by the objects to be invalidated in the **Name** field, enter its value in the **Value** field, and then specify how to match by selecting either **exact strings** or **regular expression.**

6. Optionally, in the **Search Keys** section, enter the name of a search key from the `Surrogate-Key` response-header field used by the objects to be invalidated in the **Key** field.

   > **See Also:** "Surrogate-Key Response-Header Field" on page 13-39

7. Optionally, you can preview the list of objects to be invalidated to ensure that you are removing only the objects you want to remove. To preview the list of objects:

   a. In the **Action** section, choose **Preview list of objects to be removed.**

   b. Specify the **Object Range:**

   – **From:** Enter the number representing the first object to be listed. OracleAS Web Cache begins the count of objects with the number 0.

   – **To:** Enter the number of objects to be listed.

If fewer objects than the number specified meet the invalidation criteria, OracleAS Web Cache lists the URLs for only those objects that meet the criteria.

If more objects than the number specified meet the invalidation criteria, OracleAS Web Cache lists the URLs for the number of objects requested. It also returns the total number of objects that meet the invalidation criteria. To obtain the list of URLs for addition objects, send another preview request with a different **From** number that specifies the start of the next set of objects.

c. Click **Submit.**

OracleAS Web Cache displays the Invalidation Preview Results message box, which lists the objects that meet the invalidation criteria. OracleAS Web Cache Manager lists only those objects that are valid. Although the cache may contain objects that are expired or that have been invalidated, those objects are not listed.

If the objects listed are those that you want to invalidate, continue with the next step. If they are not, modify the invalidation criteria and preview the list again.

8. In the **Action** section, select one of the following options to specify how to process invalid objects:

- **Remove immediately:** OracleAS Web Cache marks objects as invalid and then removes them immediately. A object is refreshed from the origin server when the cache receives the next request for it.

- **Remove objects no later than** *<number>* *<time>* **after submission:** OracleAS Web Cache marks objects as invalid and then refreshes them based on origin server capacity. Enter the maximum time in which the objects can reside in the cache.

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

---

9. Click **Submit.**

OracleAS 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.

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

Application Program Interfaces (APIs) for Automated Invalidation Requests

Invalidation requests can originate from a Web site's underlying application logic or from the content management application used to design Web pages.

OracleAS Web Cache ships with the following Application Program Interfaces (APIs) that you can implement:

- `jawc.jar` for a Java invalidation API

- `wxvutil.sql` and `wxvappl.sql` for a PL/SQL invalidation API

  These APIs are located in the `$ORACLE_HOME/webcache/toolkit` directory on UNIX and `ORACLE_HOME\webcache\toolkit` directory on Windows

  > **See Also:**
  >
  > - *Oracle Application Server Web Cache Invalidation API Reference* for reference information
  >
  > - `readme.toolkit.html` in the `$ORACLE_HOME/webcache/docs` directory on UNIX and `ORACLE_HOME\webcache\docs` directory on Windows for further information about the APIs

OracleAS 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.

- `EncodeBase64.java` for encoding the password using Base 64 encoding

- `invalidate.c` for C source

- `Invalidate.java` or `WCSInvalidate.java` for Java source

- `invalidate.sh` for a shell script that invalidates objects with a `telnet` session

- `invalidate.pl` for PERL source

- `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/docs` directory on UNIX and `ORACLE_HOME\webcache\docs` directory on Windows for further information about these files

Database Triggers for Automated Invalidation Requests

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/docs` directory on UNIX and `ORACLE_HOME\webcache\docs` 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

Scripts for Automated Invalidations

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 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 OracleAS 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 OracleAS 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 Object

- Example: Invalidating Multiple Objects

- Example: Invalidating a Subtree of Objects

- Example: Invalidating All Objects for a Web Site

- Example: Invalidating Objects Using Prefix Matching

- Example: Invalidating Objects Using Substring and Query String Matching

- Example: Invalidating Objects Using Search Key Matching

- Example: Propagating Invalidation Requests Throughout a Cache Cluster

- Example: Previewing Invalidation

The examples in this section require using 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 Object

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 an object 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 request 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.gif"` 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="/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-1.1">
  <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:

```
[13/Jul/2005:19:26:46 +0000] [notification 11748] [invalidation] [ecid:
 21085932167,0] Invalidation with INFO 'summary.jsp' has returned with status
 'SUCCESS'; number of objects invalidated: '2'.
.
.
.
[13/Jul/2005:19:26:46 +0000] [notification 11748] [invalidation] [ecid:
 21085932167,0] Invalidation with INFO 'summary.gif' has returned with status
 'SUCCESS'; number of objects invalidated: '14'.
```

Example: Invalidating a Subtree of Objects

The following request invalidates all objects under the /images/ directory:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.1">
  <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-1.1">
  <OBJECTRESULT>
    <ADVANCEDSELECTOR URIPREFIX="/images/" HOST="www.company.com:80"/>
    <RESULT ID="1" STATUS="SUCCESS" NUMINV="125"/>
```

```
      </OBJECTRESULT>
  </INVALIDATIONRESULT>
```

The following request invalidates all objects 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-1.1">
  <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-1.1">
   <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 Objects for a Web Site

The following request invalidates all objects under /.

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.1">
   <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-1.1">
   <OBJECTRESULT>
     <ADVANCEDSELECTOR URIPREFIX="/" HOST="www.company.com:80"/>
      <RESULT ID="1" STATUS="SUCCESS" NUMINV="17"/>
   </OBJECTRESULT>
</INVALIDATIONRESULT>
```

Example: Invalidating Objects Using Prefix Matching

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 OracleAS 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 objects 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 Objects Using Substring and Query String Matching

The following request invalidates all objects 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-1.1">
  <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:

```
[13/Jul/2005:19:26:46 +0000] [notification 11748] [invalidation] [ecid:
 21085932167,0] Invalidation with INFO 'remove-htm-under-all-post-dir has
 returned with status 'SUCCESS'; number of objects invalidated: '12'.
```

The following request invalidates all objects 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-1.1">
   <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:

```
[13/Jul/2005:19:26:46 +0000] [notification 11748] [invalidation] [ecid: 21085932
167,0] Invalidation with INFO 'remove-view-building8-10th-floor' has returned with
status 'SUCCESS'; number of objects invalidated: '3'.
```

> **See Also:** "Enhance Query String Invalidations" on page 13-37 to optimize invalidations using QUERYSTRING_PARAMETER

Example: Invalidating Objects Using Search Key Matching

The following request invalidates all objects under /pls/publicuser/, matching the following:

- Substring /pls/publicuser/!MODULE.wwpob_page.show

- HTTP request header `x-oracle-cache-user` and value `PUBLICUSER`

- `Surrogate-Key` response-header field containing a search key of `template_ id=33,31345`.

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.1">
<OBJECT>
    <ADVANCEDSELECTOR URIPREFIX="/pls/publicuser/" HOST="www.company.com:80"
      METHOD="POST">
      <OTHER NAME="SEARCHKEY" VALUE="template_id=33,31345"/>
      <HEADER NAME="x-oracle-cache-user" VALUE="PUBLICUSER"/>
      <OTHER NAME="URI" TYPE="SUBSTRING"
        VALUE="/pls/publicuser/!MODULE.wwpob_page.show"/>
    </ADVANCEDSELECTOR>
    <ACTION REMOVALTTL="0"/>
</OBJECT>
</INVALIDATION>
```

Invalidation response:

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONRESULT VERSION="WCS-1.1">
   <OBJECTRESULT>
   <ADVANCEDSELECTOR URIPREFIX="/pls/publicuser/" HOST="www.company.com:80"
      METHOD="POST">
      <OTHER NAME="SEARCHKEY" VALUE="template_id=33,31345"/>
      <HEADER NAME="x-oracle-cache-user" VALUE="PUBLICUSER"/>
      <OTHER NAME="URI" TYPE="SUBSTRING"
        VALUE="/pls/publicuser/!MODULE.wwpob_page.show"/>
      <RESULT ID="1" STATUS="SUCCESS" NUMINV="3"/>
   </OBJECTRESULT>
</INVALIDATIONRESULT>
```

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 from the Cluster Members and Properties page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Cluster Properties** > **Members and Properties**) or the Clustering page (**Properties** > **Clustering**) of OracleAS 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 OracleAS Web Cache Manager. If NAME is set to WCS_ PROPAGATE and VALUE is set to FALSE, it reads the setting specified in OracleAS 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 objects 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>
```

Inline Invalidation in HTTP Responses

Inline invalidation is implemented as part of Edge Side Includes (ESI) and provides a useful way for origin servers to "piggyback" invalidation messages on transactional responses sent to Web Cache. For instance, when a customer purchases a vegetarian cookbook on an e-commerce site, the confirmation response could contain instructions for invalidating all catalog pages related to the book, its author and vegetables. The ability to send invalidation message inline reduces the connection overhead associated with sending out-of-band invalidations and is a useful tool for ESI developers.

To configure inline invalidation:

1. In the template page, configure the HTTP response with the `Surrogate-Control` response-header field that includes `content="ESI-INV/1.0"`:

   ```
   Surrogate-Control: content="ESI-INV/1.0"
   ```

2. In the body of the same response, use the `<esi:invalidate>` tag to insert either a basic or advanced inline invalidation request.

   You can insert an inline invalidation request anywhere in the ESI template. You can insert more than one, but only the first one will be processed. The execution of the inline invalidation is blocking. That is, if the ESI template contains other ESI features, inline invalidation will be executed first.

   Basic invalidation syntax:

   ```
   <esi:invalidate [output="yes"]>
    <?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>
   </esi:invalidate>
   ```

   Advanced invalidation syntax:

   ```
   <esi:invalidate [output="yes"]>
    <?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|SEARCHKEY"
                           TYPE="SUBSTRING|REGEX"
                           VALUE="value"/>
                </ADVANCEDSELECTOR>
            </OBJECT>
        </INVALIDATION>
    </esi:invalidate>
```

> **See Also:**
>
> - "Invalidation Request Syntax" on page 13-7 for invalidation request syntax
>
> - "ESI invalidate Tag" on page 16-25 for a description of the `<esi:invalidate>` tag

Example: Using Inline Invalidation

Following is an example about an online bike shop using inline invalidation in their simple Web application. It has two CGI scripts written in Perl. `show_bike.pl` displays how many bikes of a certain model are in stock. Since it involves database query and its result remains the same until a purchase occurs, `show_bike.pl` is cached. `buy_bike.pl` is used by customers to buy a bike. Once this page is requested, `show_bike.pl` is no longer valid—an invalidation is needed.

Example 13–1 shows the code for `show_bike.pl`.

Example 13–1   show_bike.pl Code

```
#!/usr/local/bin/perl

# first, retrieve how many bikes are in stock
# and assign it to $nBikes (omitted!)

print <<END;
Content-Type: text/html
Cache-Control: private
Surrogate-Control: max-age=3600

<html>
<body>
<h1>Bike: model 2005</h1>

<p>There are $nBikes bike(s) in stock for purchase!</p>
<p>Click <a href="/cgi/buy_bike.pl">here</a> to purchase a bike.</p>

</body>
</html>
END
```

Note that `max-age=3600` informs OracleAS Web Cache to only cache this page for up to an hour.

Example 13–2 shows the code for `buy_bike.pl` with an inline invalidation request.

Example 13–2   buy_bike.pl Code with an Inline Invalidation Request

```
#!/usr/local/bin/perl

print <<END;
```

```
Content-Type: text/html
Cache-Control: private
Surrogate-Control: content="ESI/1.0 ESI-INV/1.0"

<html>
<body>
<h1>Thank you for purchasing bike model 2000.</h1>

<p>Click <a href="/cgi/show_bike.pl">here</a> to read more
about this model.</p>

<esi:invalidate>
 <?xml version="1.0"?>
 <!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
  <INVALIDATION VERSION="WCS-1.1">
    <OBJECT>
      <BASICSELECTOR URI="/cgi/show_bike.pl"/>
      <ACTION REMOVALTTL="0"/>
    </OBJECT>
 </INVALIDATION>
</esi:invalidate>

<p>Thanks again!</p>
</body>
</html>
END
```

The `ESI-INV/1.0` token in `Surrogate-Control` instructs OracleAS Web Cache to process the `<esi:invalidate>` tag.

Example 13–3 shows the browser response for `buy_bike.pl`. Because OracleAS Web Cache has already processed the inline invalidation request, the inline invalidation is not present in the response.

Example 13–3   Browser Response of buy_bike.pl

```
Content-Type: text/html
Cache-Control: private
Surrogate-Control: content="ESI/1.0 ESI-INV/1.0"

<html>
<body>
<h1>Thank you for purchasing bike model 2000.</h1>

<p>Click <a href="/cgi/show_bike.pl">here</a> to read more
about this model.</p>

<p>Thanks again!</p>
</body>
</html>
```

Debugging Tips

To facilitate debugging, the application developer can perform the following:

- Add the `Surrogate-Capability` request header that includes `"ESI-INV/1.0"`:

  ```
  Surrogate-Capability: content="ESI-INV/1.0"
  ```

When the `Surrogate-Capability` request header is added for inline invalidation, OracleAS Web Cache includes the invalidation request in the response.

- Enable the `output` attribute of the `<esi:invalidate>` tag.

  When the `output` attribute is enabled, OracleAS Web Cache includes the invalidation result in the response enclosed within comments `<!--result-->`.

Example 13–4 shows the browser response of `buy_bike.pl` when both the `Surrogate-Capability` request header is enabled for the inline invalidation and the `output` attribute of the `<esi:invalidate>` tag is enabled.

Example 13–4 Browser Response of show_bike.pl with Diagnostic Inline Invalidation Information

```
Content-Type: text/html
Cache-Control: private
Surrogate-Control: content="ESI/1.0 ESI-INV/1.0"

<html>
<body>
<h1>Thank you for purchasing bike model 2000.</h1>

<p>Click <a href="/cgi/show_bike.pl">here</a> to read more
about this model.</p>

<esi:invalidate output="yes">
 <?xml version="1.0"?>
 <!DOCTYPE INVALIDATION SYSTEM "internal:///WCSinvalidation.dtd">
  <INVALIDATION VERSION="WCS-1.1">
    <OBJECT>
      <BASICSELECTOR URI="/cgi/show_bike.pl"/>
      <ACTION REMOVALTTL="0"/>
    </OBJECT>
 </INVALIDATION>
</esi:invalidate>

<!--
<?xml version="1.0"?>
<!DOCTYPE INVALIDATIONRESULT SYSTEM "internal:///WCSinvalidation.dtd">
<INVALIDATIONRESULT VERSION="WCS-1.1">
   <OBJECTRESULT>
     <BASICSELECTOR URI="/cgi/show_bike.pl"/>
     <RESULT ID="1" STATUS="SUCCESS" NUMINV="1"/>
     </OBJECTRESULT>
</INVALIDATIONRESULT>
-->

<p>Thanks again!</p>
</body>
</html>
```

Reducing Invalidation Overhead

When OracleAS Web Cache receives an advanced invalidation request, it traverses the contents of the cache to locate the objects to invalidate. Depending on the structure and number of objects cached, it can take time for OracleAS Web Cache to invalidate 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 invalidation request rather than an advanced invalidation request to avoid cache traversal.

To send a basic invalidation request, use one of the following options:

- Select option **The single object that matches this URL** from the Invalidation page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Operations** > **Invalidation**).

- Use the Basic Content Invalidation page (**Operations** > **Basic Content Invalidation**) of OracleAS Web Cache Manager.

- Specify the BASICSELECTOR element.

For example, the following request invalidates an object 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 Content Invalidation page or specify the ADVANCEDSELECTOR element.

To send an advanced invalidation request, use one of the following options:

- Select option **Specified objects** from the Invalidation page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Operations** > **Invalidation**).

- Use the Advanced Content Invalidation page (**Operations** > **Advanced Content Invalidation**) of OracleAS Web Cache Manager.

- Specify the ADVANCEDSELECTOR element.

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 Expression** section from the Invalidation: Select Objects page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Operations** > **Invalidation**).
  - Use the **URL Expression** field from the Advanced Content Invalidation page of OracleAS Web Cache Manager (**Operations** > **Advanced Content Invalidation**).

- – 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 from the Invalidation: Select Objects page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Operations** > **Invalidation**).

  - – Use the **HTTP Method** and **POST Body Expression** fields from the Advanced Content Invalidation page of OracleAS Web Cache Manager (**Operations** > **Advanced Content Invalidation**).

  - – 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 **URL Parameters** section from the Invalidation: Select Objects page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Operations** > **Invalidation**).

  - – Use the **URL Parameters** section from the Advanced Content Invalidation page of OracleAS Web Cache Manager (**Operations** > **Advanced Content Invalidation**).

  - – Use the `OTHER` element of the `ADVANCEDSELECTOR` element with a `NAME` attribute value of `QUERYSTRING_PARAMETER`.

For the quickest invalidation, Oracle 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, use one of the following options:

- ■ Use XML message syntax:

  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`.

  4. Specify the `VALUE` attribute to use the value of `URI`, `BODY`, or `QUERYSTRING_PARAMETER`.

- ■ Use the **Substring** option for the **URL Expression** section from the Invalidation: Select Objects page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Operations** > **Invalidation**).

- ■ Use the **substring** option for the **URL Expression** field from the Advanced Content Invalidation page of OracleAS Web Cache Manager (**Operations** > **Advanced Content Invalidation**).

For example, the following request searches for any objects within `http://www.company.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 contains the substring `showPortlet.Show`.

- The HTTP `POST` body contains the substring `_language=EN-US`.

- The HTTP request headers are `x-oracle-cache-user` and `x-oracle-cache-subid` with respective values of `PORTAL` and `1`.

- The embedded URL parameters exactly match `_portlet_id` and `_provider_id`.

```
<?xml version="1.0"?>
<!DOCTYPE INVALIDATION SYSTEM
"http://www.oracle.com/webcache/90400/WCSinvalidation.dtd">
<INVALIDATION VERSION="WCS-1.1">
  <OBJECT>
    <ADVANCEDSELECTOR
URIPREFIX="/pls/portal/!PORTAL.wwpro_app_provider.execute_portlet/595897563/"
HOST="wc-cluster.company.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_PARAMETER` in an invalidation request, it can take additional processing time for OracleAS Web Cache to match and invalidate the objects.

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:

   ```
   ...
     <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="NAME_OF_QUERYSTRING_PARAMETER"/>
     <INDEXPARAM VALUE="NAME_OF_QUERYSTRING_PARAMETER"/>
    </INVALIDATIONINDEX>

    <WATCHDOG ENABLE="YES|NO"/>
```

4. Save `webcache.xml`.

5. Restart OracleAS Web Cache with the following command:

```
opmnctl restartproc ias-component=WebCache
```

The following example request invalidates all objects 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:

```
...
  <SECURITY>
  ...
  </SECURITY>

  <INVALIDATIONINDEX>
    <INDEXPARAM VALUE="building"/>
    <INDEXPARAM VALUE="floor"/>
  </INVALIDATIONINDEX>

  <WATCHDOG ENABLE="YES"/>
...
```

> **See Also:**
>
> - Table 13–1, " INVALIDATION Elements and Attributes" on page 13-9 for a description of the invalidation request syntax
>
> - "Example: Invalidating Objects Using Substring and Query String Matching" on page 13-27

Using Search Keys in Surrogate-Key Response Header and Invalidation Requests

You can base invalidation on one or more search keys used in the `Surrogate-Key` response-header field of objects in the cache. This section contains the following topics:

- Surrogate-Key Response-Header Field
- Enabling Search-Key Invalidation

Surrogate-Key Response-Header Field

The `Surrogate-Key` response-header field enables application developers to identify search key strings for a given response object. Search keys are strings that may not appear in the URL, cookies, or HTTP request headers of objects. The intent of the search keys is to provide another criteria for invalidation. In addition to the URL of objects, OracleAS Web Cache administrators can base invalidation on one or more search keys used in the `Surrogate-Key` response-header field of objects in the cache.

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

```
Surrogate-Key: search-key=("key" "key" "key" ...)
```

Usage Notes

- If `search-key` is specified in this header, then at least one search key value must be present.
- Search key values must be enclosed within quotes (`"`).
- Search key values can be of any format, such as `"key_value"` or `"key_name=key_value"`.
- The maximum number of allowed search keys is 20.
- Multiple search key values are separated by quotes.
- Space between search keys is optional.
- The search keys must remain the same.

  If search keys are replaced in a response, then OracleAS Web Cache assumes the search keys are the same.

Example Usage

The following examples show valid `Surrogate-Key` fields. The first example shows one search key of `template_id=33,31345`, and the second example shows search keys of `template_id=33,31345` and `category`.

```
Surrogate-Key: search-key=( "template_id=33,31345"  )

Surrogate-Key: search-key=("template_id=33,31345" "category")
```

The following examples show invalid `Surrogate-Key` fields. The first example shows one search key of `348` without an ending quote (`"`), and the second example shows `search-key` without any search key values.

```
Surrogate-Key: search-key=( "template_id=348 )
Surrogate-Key: search-key=( )
```

Enabling Search-Key Invalidation

To enable this feature:

1. Configure the HTTP response with the `Surrogate-Key` response-header field as follows:

   ```
   Surrogate-Key: search-key=("key" "key" "key" ...)
   ```

   > **See Also:** "Surrogate-Key Response-Header Field" on page 13-39 for a complete description of the `Surrogate-Key` response-header field

   By default, OracleAS Web Cache supports up to 20 search keys. To increase the limit:

   a. Use a text editor to open the `webcache.xml` file.

   b. Locate the `MAXSEARCHKEYSPERDOC` attribute in the `SEARCHKEYOPTIONS` element:

   ```
   <GLOBALCACHINGRULES>
           <SEARCHKEYOPTIONS ENABLE="YES" MAXSEARCHKEYSPERDOC="20"/>

           <ERRORPAGES>
   ```

   c. Modify the value to larger value.

   The following example shows a search key limit of 35.

   ```
   <GLOBALCACHINGRULES>
           <SEARCHKEYOPTIONS ENABLE="YES" MAXSEARCHKEYSPERDOC="35"/>

           <ERRORPAGES>
   ```

   d. Save `webcache.xml`.

2. Specify the search key in the invalidation request using one of the following methods:

   - Use the **Search Keys** field from the Invalidation: Select Objects page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Operations** > **Invalidation**).

   - Use the **Search Keys** section from the Advanced Content Invalidation page (**Operations** > **Advanced Content Invalidation**) of OracleAS Web Cache Manager.

   - Specify the `OTHER` element in a manual XML invalidation request to use the `ADVANCEDSELECTOR` element, and specify the `NAME` and `VALUE` attributes to use `SEARCHKEY` and the search key value, respectively.

     > **See Also:**
     >
     > - "Submitting Advanced Invalidation Requests" on page 13-19 for instructions on using OracleAS Web Cache Manager
     >
     > - Table 13–1, " INVALIDATION Elements and Attributes" on page 13-9 for a description of manual invalidation request syntax
     >
     > - "Example: Invalidating Objects Using Search Key Matching" on page 13-28 for an example scenario

14

Monitoring Cache Trends with Statistics

> **See Also:** *Oracle Application Server Performance Guide* for performance tuning tips

This chapter describes how to gather performance statistics to monitor cache operation trends.

This chapter contains these topics:

- Monitoring OracleAS Web Cache Health
- Configuring End-User Performance Monitoring
- Gathering End-User Performance Data
- Viewing Cache Performance Statistics
- Viewing Origin Server Performance Statistics

Note that while this chapter focuses on the use of OracleAS Web Cache Manager, you can use Oracle Enterprise Manager 10*g* Application Server Control Console and Oracle Enterprise Manager 10*g* Grid Control Console to monitor many of the same performance statistics described in this chapter. See Chapter 6 and the *Oracle Application Server Administrator's Guide* for further information about using Oracle Enterprise Manager 10*g* to monitor Oracle Application Server components.

If after monitoring metrics from either OracleAS Web Cache Manager or Enterprise Manager you need additional performance metrics, point your browser to the following URL:

```
http://web_cache_hostname:web_cache_stats_port
```

The default port is 9402. This URL takes you to the Oracle Web Cache Internal Diagnosability Monitor page, which provides additional information about hits and misses.

> **See Also:** "Task 8: Configure OracleAS Web Cache with Operations Ports" on page 8-22 for further information about configuring the statistics monitoring port

> **Note:** After making a Statistics port property change in the Operations Ports page (**Ports** > **Operations Ports**) in OracleAS Web Cache Manager, ensure that the `cache` server process is restarted. If it is not, then the Monitoring pages (Web Cache Statistics, Health Monitor, Origin Server Statistics, and Popular Requests) report the following error:
>
> ```
> Failure obtaining statistics from Web Cache cache
> server: error in connect. Please check that the
> cache server is up and running.
> ```
>
> See "Task 12: Restart OracleAS Web Cache" on page 8-42 for instructions on applying configuration changes and restarting the `cache` server process with command-line tools or OracleAS Web Cache Manager.

Monitoring OracleAS Web Cache Health

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

To monitor overall cache health from OracleAS Web Cache Manager:

1. In the navigator frame, select **Monitoring** > **Health Monitor**.

   The Health Monitor page appears in the right pane.

2. From the **For Cache** list, select the cache and click **View**. If you have not configured a cache cluster, this field displays the cache to which you are connected.

3. From the **Auto Refresh** list, select the frequency at which you want the statistics refreshed and click **Set**.

   If you select **Never**, then the page will not be refreshed again. If you want the statistics refreshed now, click **Refresh Now**.

Table 14–1 through Table 14–3 describe the statistics for this page.

Table 14–1 describes the general statistics.

Table 14–1   OracleAS Web Cache Health Monitor: General Statistics

Statistic	Description
Current Time	The time when this page was generated
Current Web Cache Start Time	The time when OracleAS Web Cache was started
Time Since Start	The length of time that OracleAS Web Cache has been operating since it was started. Time is denoted in *days/hours/minutes/seconds*.
Total Number of Requests Served by Current Web Cache	Accumulated number of requests that OracleAS Web Cache has served since it was started
	See Also: "Viewing Cache Performance Statistics" on page 14-8 to view detailed statistics for OracleAS Web Cache

Table 14–2 describes the statistics shown in the **Requests Served by Origin Servers** table, which lists the number of requests served to the selected cache by the origin servers.

Table 14–2   Requests Served by Origin Servers

Statistic	Description
Requests Served by Origin Servers	Name of the **origin server** and the port number from which the origin server is listening for OracleAS 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	The status of the origin server at the time that OracleAS Web Cache last attempted to communicate with that origin server. (OracleAS Web Cache attempts to reach the origin server only for specific purposes, such as retrieving responses for a cache miss.)
	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, OracleAS Web Cache continues to forward requests. If this is not the last server, no new requests will be sent to server. However, OracleAS Web Cache will poll the server until it is back online.
Since	Date and time when the origin server was last known to be up or down.
Total Requests Served	Number of client requests, such as Web browser requests, resolved by this origin server.
Average Latency	Average amount of time for the client requests to be resolved.

Table 14–3 describes the statistics in the **Serving Requests/Second Now** table.

Table 14–3   Serving Requests/Second Now Statistics

Statistic	Description
Stale Serves from Current Web Cache	Objects in the cache that have expired or that have been invalidated, but have not yet been refreshed from the origin servers
Fresh Serves from Current Web Cache	Objects in the cache that are still valid
	The health bar provides a graphical view of the number of client requests resolved for each second.

Configuring End-User Performance Monitoring

Using Oracle Enterprise Manager 10*g* Grid Control Console or OracleAS Web Cache Manager, you can monitor the response time of your applications by viewing information about how quickly the responses are delivered to the end users. From the time a user enters the Web site until they exit, you can monitor which URLs they view and view reports about the response times the end user has experienced.

You can view this information using Grid Control Console. End-user performance monitoring is part of the Application Service Level Management feature.

To configure OracleAS Web Cache for end-user performance monitoring, ensure that the following prerequisites are fulfilled:

- Configure a network load balancer for a deployment supporting multiple OracleAS Web Cache servers. The network load balancer enables OracleAS Web Cache to receive requests from the same clients.

- Ensure that software for Grid Control Console is configured on the network.

You can enable end-user performance monitoring for a specific cache or for a site. The basic steps are:

1. Follow instructions in the *Oracle Enterprise Manager Advanced Configuration* to create a Web Application target in Grid Control Console.

A **Web Application** is an Enterprise Manager target that consolidates all the components of your Web application and displays the availability and performance of the application

2. Using Oracle Enterprise Manager 10*g* Application Server Control Console or OracleAS Web Cache Manager, enable End-User Performance Monitoring for a cache or a site. See "Enabling End-User Performance Monitoring" on page 14-4 for instructions.

3. From Grid Control Console, navigate to the Web Application home page, and click the **Administration** tab to display the Web Application Administration page. From Grid Control Console, on the Configure Web Application Web Caches page, start the collection of data and specify an interval for the collection.

4. Click **Configure Web Application Web Caches** to start the collection of data and specify an interval for the collection.

> **See Also:**
>
> ■ "Overview of Configuring End-User Response Time Monitoring" in Enterprise Manager Online Help for instructions
>
> ■ *Oracle Enterprise Manager Advanced Configuration*

Enabling End-User Performance Monitoring

To enable end-user performance monitoring for a cache or a site in Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **End-User Performance Monitoring**.

> **See Also:** "Configuring End-User Performance Monitoring" in Enterprise Manager Online Help for instructions

To enable end-user performance monitoring for a cache or a site in OracleAS Web Cache Manager:

1. In the navigator frame of OracleAS Web Cache Manager, select **Logging and Diagnostics** > **End-User Performance Monitoring**.

2. You can enable monitoring for a particular cache or for an entire site.

   ■ To enable monitoring for a particular cache, in the End-User Performance Monitoring page, select the cache from the **Cache-Specific End-User Performance Monitoring** section. Then, click **Enable**.

   ■ To enable monitoring for the entire site, in the End-User Performance Monitoring page, select the site from the **Site-Specific End-User Performance Monitoring** section. Then, click **Enable.**

   Note the following:

   ■ If you enable monitoring for a specific cache, but you disable monitoring for a site, monitoring is performed for all the sites in that cache node except for the site for which it is disabled.

   If a request does not match any of the sites for which monitoring is enabled, the request will not be monitored.

   ■ If you disable monitoring for a cache, monitoring is not performed for any of the sites of that cache node.

   ■ For a site, you can specify end-user performance monitoring rules.

When end-user performance monitoring is enabled for a cache, Grid Control Console and OracleAS Web Cache Manager monitor all requests that are routed through the cache. When end-user performance monitoring is enabled for a site, Grid Control Console and OracleAS Web Cache Manager monitor all of the Web pages on your Web site and provide you with information about how the pages are responding to customer requests. Using Grid Control Console, you can sort this information by URL, region, domain, visitor, and Web server.

3. Optionally, select URLs to monitor or not monitor by creating site-specific monitoring rules as described in "Selecting URLs to Monitor" on page 14-5.

4. Configure OracleAS Web Cache to use the End-User Performance Monitoring access log format, as described in "Configuring Access Logs" on page 15-22.

5. Start end-user performance monitoring:

   a. From Grid Control Console, on the Configure Web Application Web Caches page, click **Collecting**.

   b. In the **Interval (minutes)** column, enter the interval at which to collect performance data.

   c. Click **OK**.

Selecting URLs to Monitor

For each site, you can further refine which Web pages are monitored, by specifying regular expressions or path substrings.

In the Set Up End-User Performance Monitoring page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **End-User Performance Monitoring**), click **Filtered** or **All** to configure the end-user performance monitoring rules for the selected site.

> **See Also:** "Configuring End-User Performance Monitoring" in Enterprise Manager Online Help for instructions

In OracleAS Web Cache Manager, take the following steps:

1. In the navigator frame, select **Logging and Diagnostics** > **End-User Performance Monitoring**.

2. In the **Site-Specific End-User Performance Monitoring** section of the End-User Performance Monitoring page, select the site, then click **View.**

3. To add a site-specific rule, click **Create Site-Specific Rule.**

   The Create End-User Performance Monitoring Rule page is displayed.

4. In the **URL Expression** field, specify the URL expression. Note the following information, which is dependent on the **Expression Type** selected:

   - **Path Substring:** Enter a path substring.

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

     For example, if you enter a substring of down, any of the following URLs match the expression:

     ```
     http://www.company.com/wireless/downloads/index.htm
     ```

```
http://www.company.com/support/updown.htm
http://www.company.com/support/shutdown_gdlines.htm
```

- **Regular Expression:** Enter the regular expression. Remember to use "^" to denote the start of the URL and "$" to denote the end of the URL.

  Regular expression syntax 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:

  – 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 (*).

  For example, to monitor all URLs beginning with `/machine/doc` and ending in `.gif`, enter the following expression:

  ```
  ^/machine/doc/.*\.gif$
  ```

5. From the list in the **Expression Type** column, select one of the following options:

   – **Path Substring:** Applies the monitoring rule to objects matching a substring of a path

   – **Regular Expression:** Applies the monitoring rule to objects matching **regular expression** syntax

6. For **Monitoring Policy**, select **Monitor** or **Don't Monitor** for the rule.

7. Click **Submit.**

8. To create another monitoring rule, click **Edit, Insert Above,** or **Insert Below.** Then, repeat Steps 4 through 7.

9. If you have more than one monitoring rule, you can use the **Move Up** and **Move Down** buttons on the End-User Performance Monitoring page to prioritize the rules. Higher priority rules are processed first. The highest priority rule appears first in the list. The lower the number listed in the priority column, the higher the priority. For example, the highest priority rule is assigned a priority of one (1).

   For a regular expression that matches objects or a subset of objects that you do not want to monitor, give the rule that specifies **Don't Monitor** a higher priority than the rule that specifies **Monitor**. For example, if you want all URLs containing `/cec/cstage?ecaction=ecpassthru` to be monitored except for `/cec/cstage?ecaction=ecpassthru2`, you would enter the rules in the following order:

   a. `^/cec/cstage\?ecaction=ecpassthru2,Don't Monitor`

   b. `^/cec/cstage\?ecaction=ecpassthru.*,Monitor`

If the order were reversed, all objects starting with
`/cec/cstage?ecaction=ecpassthru` would be monitored, including
`/cec/cstage?ecaction=ecpassthru2`.

Gathering End-User Performance Data

Using Oracle Enterprise Manager 10*g* Grid Control Console or OracleAS Web Cache
Manager, you can monitor the response time of your applications by viewing
information about how quickly the responses are delivered to the end users. From the
time a user enters the Web site until they exit, you can monitor which URLs they view
and view reports about the response times the end user has experienced.

You can view this information using Grid Control Console. End-user performance
monitoring is part of the Application Service Level Management feature.

> **See Also:** "Configuring End-User Performance Monitoring" on
> page 14-3 for information about configuring end-user performance
> monitoring.

Alternatively, you can use OracleAS Web Cache Manager to access the data, but this
feature is for evaluation purposes only and is currently unsupported. For more
complete analysis and reporting, use the fully-supported Grid Control Console
Application Service Level Management.

With OracleAS Web Cache Manager, you can view the output in HTML or write it to a
comma-delimited file, which can be imported into spreadsheet tools for viewing and
analysis. The output provides information about the URLs or domains accessed and
the average, minimum, and maximum latency of the requests to each URL.

Note that generating the output may take several minutes or hours to complete,
depending on the log file size. Because this operation is processed on the OracleAS
Web Cache server, perform this operation during non-peak hours to maintain the
performance of your cache.

To use this feature, take the following steps:

1. Configure the cache for end-user performance monitoring, as described in
    "Configuring End-User Performance Monitoring" on page 14-3.

2. In the navigator frame of OracleAS Web Cache Manager, select **Monitoring** >
    End-User Performance Analysis.

    The End-User Performance Analysis page appears.

3. In the **Specify Access Log File for Analysis** field, specify the full file specification
    for the access log that you want to analyze.

4. Click **Submit.**

    OracleAS Web Cache Manager displays the Analyzing End-User Performance
    dialog box, which provides progress messages. Do not close this dialog box.

5. When the operation is complete, the Analyzing End-User Performance dialog box
    displays methods of viewing the results. Select one of the following:

    –   To view the output in HTML, select **View output in HTML.** The output is
        displayed in a browser window. To specify how the information is sorted,
        select one of the following from the links in the browser window:

        *   **By URL:** Access log entries are sorted by URL.

        *   **By Domain:** Access log entries are sorted by domain.

    *   **By URL (Hourly):** Access log entries are sorted by hour, then by URL.

    *   **By Domain (Hourly):** Access log entries are sorted by hour, then by domain.

   –   To download the output to a comma-delimited (`.csv`) file, select one of the following:

    *   **By URL:** Access log entries are sorted by URL.

    *   **By Domain:** Access log entries are sorted by domain.

    *   **By URL (Hourly):** Access log entries are sorted by hour, then by URL.

    *   **By Domain (Hourly):** Access log entries are sorted by hour, then by domain.

You are prompted to save or open the file. You can view the file in any application, such as Microsoft Excel, that supports comma-delimited files.

Viewing Cache Performance Statistics

To monitor performance for a cache in OracleAS Web Cache Manager:

1.   In the navigator frame, select **Monitoring > Web Cache Statistics.**

    The Web Cache Statistics page appears.

2.   From the **For Cache** list, select the cache and click **View.**

3.   From the **For Site** list, select the Web site for which to view statistics. Click **View.**

4.   From the **Auto Refresh** list, select the frequency at which you want the statistics refreshed, and click **Set.**

    If you select **Never**, then the page will not be refreshed again. If you want the statistics refreshed now, click **Refresh Now.**

Table 14–4 through Table 14–8 describe the statistics on this page.

Table 14–4 describes the general statistics about the cache.

Table 14–4   OracleAS Web Cache Statistics: General Statistics

Statistic	Description
Current Cache Start Time	The time when OracleAS Web Cache was started or restarted
Time Since Start	The length of time that OracleAS 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 14–5 describes the statistics provided by the **Cache Overview** table, which provides an overview of cache performance.

Table 14–5   OracleAS Web Cache Statistics: Cache Overview

Statistic	Description
Number of Objects in Cache	Number of objects stored in OracleAS Web Cache, plus the number of objects in transit through the cache. The number includes objects that have expired or have been invalidated but that have not been deleted from the cache.
Size of Objects in Cache	Size of the *contents* of the objects currently in the cache. This size does not reflect the total size needed to cache the objects because it does not include object header information or object overhead.
Total Bytes Served	Total number of bytes served to clients, such as browsers
Total Bytes Saved by Compression	Additional bytes that would be sent to clients if in-cache compression is disabled
Current Number of Open Connections	Current number of incoming open connections to the OracleAS Web Cache server and outgoing open connections to the origin servers. You can adjust the limit of connections in the Resource Limits page (**Properties** > **Resource Limits**).
Configured Maximum Cache Size	The maximum cache size. You can adjust the maximum size of the cache in the Resource Limits page (**Properties** > **Resource Limits**).
Current Allocated Memory	The physical size of the cache. The physical size of the cache is the amount of data memory allocated by OracleAS Web Cache for cache storage and operation. This number is always smaller than the process size shown by operating system statistics because the OracleAS 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-five percent of the **Configured Maximum Cache Size**. This number is usually larger than the **Current Allocated Memory**. See "Cache Memory" on page 8-13 for more information about the relationship among Current Action Limit, Configured Maximum Cache Size, and Current Allocated Memory.

If you have a cache cluster, the **Detailed Statistics** link appears. This link displays the Web Cache Detailed Statistics page, which is described in "Viewing Detailed Statistics for a Cache Cluster" on page 14-10.

Table 14–6 describes the **Requests Served** table. This table provides information about the number or percentage of requests OracleAS Web Cache: has served on average for each second for the last ten seconds (**Recent** column), has served since it was started (**Since Start** column), and has served since the metrics were reset (**Since Reset** column).

Table 14–6   OracleAS Web Cache Statistics: Requests Served

Statistic	Description
Total Requests Served	Accumulated number of browser, peer cache, and ESI requests that OracleAS Web Cache has served since it was started or restarted
Average Requests Served	Average number of client, peer cache, and ESI requests served for each second
Fresh Hits	Percentage of client requests resolved by objects in the cache This percentage should be high, except when objects are being invalidated.
Stale Hits	Percentage of client requests resolved by objects that have expired or have been invalidated, but for which updated version have not yet been retrieved from the origin servers. As objects are invalidated or expired, the percentage of stale hits will increase. The percentage will decrease as OracleAS Web Cache retrieves updated content from the origin servers. If the percentage does not decrease, it could indicate a bottleneck on the origin servers.

Table 14–6   (Cont.) OracleAS Web Cache Statistics: Requests Served

Statistic	Description
Cacheable Misses	Percentage of client requests that were not served directly by OracleAS Web Cache, but were served by OracleAS Web Cache after it fetched the content from the origin server. In a cache cluster, this number includes requests from peer caches.
Noncacheable Misses	Percentage of client requests for non-cacheable objects that were not served by OracleAS Web Cache.
Refreshes	Percentage of objects that OracleAS Web Cache has refreshed from the origin servers
Compressed Hits	Percentage of total requests served by the cache in compressed form
Compressed Misses	Percentage of total requests retrieved from the origin server and compressed by OracleAS Web Cache before serving

Table 14–7 describes the Cache Errors table. The Cache Errors table provides metrics about the error pages served since OracleAS Web Cache was started (**Since Start** column) or when the metrics were reset (**Since Reset** column).

Table 14–7   OracleAS Web Cache Statistics: Errors Served

Statistic	Description
Network Error	Number of error pages that OracleAS Web Cache has served to clients due to a network error
Site Busy Error	Number of error pages that OracleAS Web Cache has served to clients due to a busy Web site error
ESI Uncaught Exception	Number of errors that OracleAS Web Cache has returned to a client when an uncaught exception occurred in the cache during ESI parsing or processing
ESI Default Fragment	Number of error pages that OracleAS Web Cache has served when an uncaught exception occurs in the application during ESI parsing or processing

Table 14–8 describes the Invalidations table, which provides metrics on the invalidation requests served since OracleAS Web Cache was started (**Since Start** column) or when the metrics were reset (**Since Reset** column).

Table 14–8   OracleAS Web Cache Statistics: Invalidations

Statistic	Description
Total Invalidation Requests	The number of invalidation requests processed
Total Invalidation Objects	The total number of objects invalidated

Viewing Detailed Statistics for a Cache Cluster

If you have a cache cluster, you can view detailed statistics for the caches in that cluster.

To view these detailed statistics in OracleAS Web Cache Manager:

1. In the navigator frame, select **Monitoring** > **Web Cache Statistics**.

   The Web Cache Statistics page appears.

2. Click the **Detailed Statistics** link located at the top of the **Cache Overview** table.

   The Detailed Web Cache Statistics page appears.

3. From the **For Cache** list, select the cache and click **View.**

If you select a specific cache, then information presented in the Web Cache Statistics page displays, which is described in "Viewing Cache Performance Statistics" on page 14-8. In addition, the **Cache Overview** table in this page shows metrics for **owned content** and **on-demand content**.

If you select **All Caches**, then Cluster Web Cache Statistics view of this page displays. Table 14–9 describes the statistics for this view.

4. From the **For Site** list, select the Web site for which to view statistics. Click **View**.

5. From the **Auto Refresh** list, select the frequency at which you want the statistics refreshed, and click **Set**.

   If you select **Never,** then the page will not be refreshed again. If you want the statistics refreshed now, click **Refresh Now**.

Table 14–9   Cluster Web Cache Statistics

Statistic	Description
Number of Objects in All Caches	Total number of objects stored in all caches in the cache cluster, plus the number of objects in transit through a cache. The number includes objects that have expired or have been invalidated, but that have not been deleted from a cache.
Size of Objects in All Caches	Size of the *contents* of the objects currently in all caches in the cache cluster. This size does not reflect the total size needed to cache the objects because it does not include object header information or object overhead.
Total Requests Served by All Caches	Total accumulated number of client, peer cache, and ESI requests that all caches in the cache cluster have served since each cache was started or restarted.
Current Hit Rate in All Caches	Percentage of total requests resolved by objects in all caches in the cache cluster
Current Miss Rate in All Caches	Percentage of total requests to all caches in the cache cluster that were retrieved from the origin server

Viewing Origin Server Performance Statistics

To monitor origin server performance in OracleAS Web Cache Manager:

1. In the navigator frame, select **Monitoring** > **Origin Server Statistics**.

   The Origin Server Statistics page appears.

2. From the **For Cache** list, select the cache and click **View.**

   If you have not configured a cache cluster, this field displays the current cache (the cache to which you are connected.)

3. From the **Auto Refresh** list, select the frequency at which you want the statistics refreshed and click **Set.**

   If you select **Never**, then the page will not be refreshed again. If you want the statistics refreshed now, click **Refresh Now.**

Table 14–10 through Table 14–12 describe the statistics for this page.

Table 14–10 on page 14-12 describes the general information about the origin servers, that is, application Web servers or proxy servers, displayed on this page.

Table 14–10   Origin Server Statistics

Statistic	Description
Origin Server: hostname	Name of the origin server and the port number from which the server is listening for OracleAS Web Cache requests.
Origin Server: 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**: The status of the origin server at the time that OracleAS Web Cache last attempted to communicate with that origin server. (OracleAS Web Cache attempts to reach the origin server only for specific purposes, such as retrieving responses for a cache miss.)
	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, OracleAS 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: Date and time when the origin server was last known to be up or down
Completed Requests	**number/sec**: Current number of requests that the origin server has processed for each second. The number is the average for each second during the last ten-second interval.
	max/sec: Maximum number of requests that the origin server has processed for each second during the last ten-second interval.
Latency	**avg this interval**: Average time this origin server has taken to process and reply to requests which it has received from this OracleAS Web Cache, over the last 10 seconds.
	avg since start: Average time this origin server has taken to process and reply to requests which it has received from this OracleAS Web Cache, since OracleAS Web Cache has been started.
Load	**now**: Current number of connections from OracleAS Web Cache that the origin server has open. The number is the average for each second during the last ten-second interval.
	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 Origin Servers page (**Origin Servers, Sites, and Load Balancing** > **Origin Servers**).
Active Sessions	Active sessions are based on the Internal-Tracking session binding mechanism configured. See "Bind a Session to an Origin Server" on page 8-39.
	now: Current number of active sessions from OracleAS Web Cache to the origin servers.
	max: Maximum number of active sessions that the origin server has had open at one time.

Table 14–11 on page 14-13 describes the type of errors listed in the Errors Served table, which lists the following for each type of errors served:

■ **Number this second:** Current number error pages that OracleAS Web Cache is serving

■ **Total:** Total number of error pages that OracleAS Web Cache has served since the cache was last restarted

Table 14–11   Errors Served

Type	Description
Network Error	Error pages that OracleAS Web Cache has served to clients due to a network error
Site Busy Error	Error pages that OracleAS Web Cache has served to clients due to a busy Web site error
ESI Uncaught Exception	Number of errors that OracleAS Web Cache has returned to a client when an uncaught exception occurred during ESI parsing or processing
ESI Default Fragment	Number of error pages that OracleAS Web Cache has served to a client when an uncaught exception occurs in the application during ESI parsing or processing

Table 14–12 describes the statistics in the Origin Server Backlog table.

Table 14–12   Origin Server Backlog Statistics

Type	Description
Now	Current number of requests for the cache that are awaiting processing by the application Web server.
Max	Maximum number of requests for the cache that are awaiting processing by the application Web server.

15

Using Diagnostics Tools

This chapter explains how to use the tools provided by OracleAS Web Cache for diagnostic purposes.

This chapter contains these topics:

- Listing Popular Requests and Cache Contents
- Evaluating Event Logs
- Displaying Diagnostic and Event Log Information in the HTML Body or Server Response-Header Field
- Evaluating Access Logs
- Rolling Over Event and Access Logs

Listing Popular Requests and Cache Contents

With Oracle Enterprise Manager 10*g* Application Server Control Console or OracleAS Web Cache Manager, you can view a list of the most popular requests and a list of the contents of the cache, generating the following types of lists:

- A list of the URLs of the most popular requests received by the cache since the cache was last started

  Popularity is calculated using two factors: how many times the request was made and how recently the requests were made. You can specify: only objects stored in the cache, only objects not stored in the cache, or all requests received by the cache.

- A list of the URLs of the objects currently in the cache

  OracleAS Web Cache writes the list of URLs to a file. You can use this list to verify that the caching rules are caching the objects that you want cached.

Listing Popular Requests

To view the list of URLs of the most popular requests with Application Server Control Console, navigate to **Web Cache Home** page > **Performance** tab > **All Sites** section > **Popular Requests** link.

> **See Also:** "Listing Popular Browser Requests" in Enterprise Manager Online Help for instructions

To view the list of URLs of the most popular requests with OracleAS Web Cache Manager:

1. In the navigation pane, select **Monitoring** > **Popular Requests**.

The Popular Requests page appears in the right pane.

2. From the **For Cache** list, select a cache and click **View**. (More than one cache is listed only if you configured a cache cluster.)

3. In the **List Most Popular Requests** area, for **Number of Objects**, enter the maximum number of URLs you want displayed.

4. For **Filter Objects,** select one of the following options:

   – **All**: All requests received by the cache.

   – **Cached**: Only those requests stored in the cache.

   – **Not Cached**: Only those requests not stored in the cache.

5. Click **Update.**

OracleAS Web Cache Manager displays a table containing the list of URLs of requests 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).

- **Cached**: Whether or not the object is cached. The possible values are:

  – Yes: The object is cached.

  – Yes, but expired: The object is cached, but it is expired. (To lessen the performance impact of invalidation and expiration, OracleAS Web Cache serves some stale objects until the origin servers have the capacity to refresh them.)

  – No: The object is not cached.

- **Cache Reason**: The reason that the object is cached or not cached. Possible values are:

  – BY_CACHING_RULE: Cached or not cached because of the caching rule.

  – BY_HTTP_HEADERS: Cached or not cached because of information in the HTTP header.

  – BY_PRESEEDING: Cached because the objects were preseeded for use by end-user performance monitoring.

  – BY_REFERENCE_TTL: Cached because of the nonzero value of the reference TTL (time-to-live parameters) specified in the ESI tag.

  – BY_SURROGATE_CONTROL_HEADER: Cached or not cached because of information in the Surrogate-Control response header.

  – BY_VALIDATION: Cached because of the ETag response header.

  – BY_X_ORACLE_HEADERS: Cached or not cached because of information in the `X-Oracle-Cache` response header.

  – NC_COOKIE_MISMATCH: Not cached because the response contains a cookie that is not present in the request or that has a different value than the same cookie in the request.

- – NC_HTTP_RESPONSE_CODE: Not cached because the response was not an HTTP status code of 200.

- – NC_NO_DIRECTIVE: Not cached because no directive or rule has stated that the object should be cached.

- – NC_POST_BODY_TOO_BIG: Not cached because the POST body was too large to be cached.

- – NC_REQUEST_METHOD: Not cached because the request method was other than a GET or POST.

- – NC_SIZE_TOO_LARGE: Not cached because the object is larger than the size specified as the Maximum Object Size.

- – NC_WITH_QUERYSTRING: Not cached because the request contains a query string but the request did not match any caching rules.

- ■ **Caching Rule**: If a caching rule is associated with the object, this column displays a link to the Cacheability Rule Details page. That page shows the regular expression and site information for the URL.

- ■ **Compressed:** Whether or not the object is compressed.

Listing All Contents

Using OracleAS Web Cache Manager, you can also generate a list of the URLs of all of the objects currently stored in the cache. Application Server Control Console does not support this feature.

1. In the navigation pane, select **Monitoring** > **Popular Requests**.

   The Popular Requests page appears in the right pane.

2. From the **For Cache** list, select a cache and click **View**. (More than one cache is listed only if you have a cache cluster.)

3. For **List All Contents in Cache**, click **Export to File**.

   The Export Cache Contents dialog box appears. It lists the file to which OracleAS Web Cache will write the URLs. The file is written to the OracleAS Web Cache log directory and is named `webcache_contents.txt`.

4. To write the list to a different location, enter a complete file specification in the text box.

5. Click **Submit.**

OracleAS Web Cache writes the list of URLs to the text file you specified. Each time you generate the list, OracleAS 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 Jul 20 11:47:03 2005
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 Jul 20 13:01:24 2005
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

OracleAS Web Cache events and errors are stored in an **event log**. The event log can help you determine which objects have been inserted into the cache. It can also identify listening port conflicts or startup and shutdown issues. By default, 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:

- Oracle-ECID Request-Header Field
- Format of the Event Log File
- Configuring Event Logs
- Event Log Examples

> **See Also:** Chapter 17 for descriptions of the event log messages

Oracle-ECID Request-Header Field

The `Oracle-ECID` request header is used to track requests as they move through the Oracle Application Server architecture. This information is especially useful for diagnostic purposes. Because OracleAS Web Cache is the initial receiver of client requests, it sets the request header before forwarding a cache miss to an origin server. The `Oracle-ECID` request header takes the following format:

```
Oracle-ECID: request_id, sequence_number
```

In the format, `request_id` is a 64-bit unique integer for the request, and `sequence_number` is the hop number of the request as it passes through Oracle Application Server. OracleAS Web Cache typically assigns an initial sequence number of 0 to a request. As a request passes from OracleAS Web Cache to other Oracle Application Server components, the request ID remains constant, but the sequence number increments with each hop.

You can configure OracleAS Web Cache to log the request ID and sequence number from the `Oracle-ECID` request header in the event and access logs. To display the `Oracle-ECID` request header in the logs, you enable **Include Request Details** for the event log, and select the `x-ecid` field for the access logs. The `x-ecid` field is provided by default with the Enhanced CLF (ECLF), Enhanced Combined Log Format, and End-User Performance Monitoring Format. Additionally, you can configure Oracle HTTP Server to log the `Oracle-ECID` request header information, enabling you to correlate events at different Oracle Application Server stops for the same request.

OracleAS Web Cache also includes `Oracle-ECID` request header information whenever you configure to display diagnostic information in the `Server` response-header field or the HTML response body.

> **See Also:** "Displaying Diagnostic and Event Log Information in the HTML Body or Server Response-Header Field" on page 15-11 for further information about configuring diagnostic output in the `Server` response-header field or the HTTP response message that includes `Oracle-ECID` information

Format of the Event Log File

Event messages are written using either of the following formats:

- The default format displays a typical log message for each instance of an event. It displays the request ID and sequence number of the `Oracle-ECID` request header so that you can easily find the matching request detail line.

  [*Timestamp*][*severity message_id*][ecid: *request_id*, *serial_number*] *Event_Log_ Message*

- The request detail format is displayed when you enable **Include Request Details**. The field [detail] indicates that the event is for a request. This format is logged the first time an event is logged for a request. In addition to the IP address, site name, and URL of the request, the ID and sequence number of the `Oracle-ECID` request header is logged. The `Oracle-ECID` request header is used to track requests.

  [*Timestamp*][*module_name*][detail][ecid: *request_id*, *hop_number*] [client: *IP_ address*] [host: *site*] [url: *URL*]

Table 15–1 describes the fields for the default event log format.

Table 15–1   Event Log Format

Fields	Description
[*Timestamp*]	Date when the event occurred. Time is displayed in either local or Greenwich Mean Time.
[*severity message_id*]	Severity level and message ID of event log message. Severity level can be one of the following:
	■ `internal error`: An event that occurs due to a bug
	■ `alert`: An event that affects the entire OracleAS Web Cache instance and causes a lack of service in subsequent request/response transaction
	■ `error`: An event that causes a lack of service only within the request/response transaction
	■ `warning`: An event which is abnormal or potentially a sign of some problem
	■ `notification`: An event that provides current status
	■ `trace`: An event that indicates a trace activity
	■ `debug`: An event that is a trace activity intended for Oracle Support Services and development
[ecid: *request_id*, *serial_number*]	Request ID and hop-sequence number from the `Oracle-ECID` request header
Event_Log_Message	Event message

Table 15–2 describes the fields for the request detail format.

Table 15–2   Event Log Format for Request Details

Fields	Description
`[Timestamp]`	Date when the event occurred. Time is either displayed in local or Greenwich Mean Time.
`[detail]`	Request detail event
`[ecid: request_id, serial_number]`	Request ID and sequence number from the `Oracle-ECID` request header
`[client: IP_address]`	IP address of the client that made the request
`[host: site]`	Site name of the request
`[url: URL]`	URL of the request

Configuring Event Logs

To configure the event log settings with Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Logging**.

> **See Also:**   "Configuring Event Logs" in Enterprise Manager Online Help for instructions

To configure event log settings with OracleAS Web Cache Manager:

1.   In the navigator frame, select **Logging and Diagnostics** > **Event Logs.**

     The Event Logs page appears.

2.   Specify cache-specific event log settings:

     a.   From the Cache-Specific Event Log Configuration table, select a cache, and then click **Edit Selected.**

     The Edit Cache-Specific Event Log Configuration dialog box appears.

     b.   In the **Directory** field, enter the directory in which to write event logs.

     The default is `$ORACLE_HOME/webcache/logs` on UNIX and `ORACLE_HOME\webcache\logs` on Windows.

     c.   In the **Buffering** field, select **Enabled** to enable buffered logging or **Disabled** to disable buffered logging.

     With buffered logging, OracleAS Web Cache writes to the event log after the buffer is full. The buffer size is set to 2048 bytes. When the limit is reached, OracleAS Web Cache writes buffered events to the event log file.

     Oracle recommends disabling buffering when you need to see the event log results immediately, such as when you are debugging OracleAS Web Cache.

     d.   If buffering is enabled, in the **Flush Interval** field, enter the maximum threshold, in seconds, when contents of the buffer are written to the event log file.

     The default is 10 seconds. When the maximum threshold is reached, OracleAS Web Cache writes buffered events to the event log file. Even if the buffer is not full, the event log is updated at least every 10 seconds. Oracle recommends not changing the default unless you need to lower the interval to see results more frequently.

e. From the **Verbosity** list, select the needed level of detail for the event log. The levels are described in Table 15–3.

Table 15–3   Verbosity Levels

Level	Description
WARNING	Provides abnormal-operation events.
NOTIFICATION	Provides normal-operation events, such as startup and shutdown. This is the default.
TRACE	Provides events for debugging caching rules.
DEBUG	Provides detailed events for troubleshooting. This level is intended for Oracle Support Services.

f. Click **Submit.**

3. Set the global event log settings:

a. From the Global Event Log Configuration table, click **Edit.**

The Edit Global Event Log Configuration dialog box appears.

b. In the **File Name** field, enter a name for the event log file.

The default file name is `event_log`.

c. From the **Time Style** list, select either **LOCAL** or **GMT** (Greenwich Mean Time) to modify the time stamp style associated with entries in the event log file.

Note: Oracle 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, OracleAS Web Cache has no mechanism to improve the performance of the conversion process.

---

d. In **Include Request Details**, select **Yes** to enable the logging of request details or **No** to disable logging of requests.

Oracle recommends selecting **No** if either of the following conditions apply:

–   You are concerned about the performance impact of event log entries for request details

–   You are running OracleAS Web Cache in a standalone environment without Oracle HTTP Server

See Also:

■   "Format of the Event Log File" on page 15-5 for further information about how request details are logged

■   Appendix B for information about running OracleAS Web Cache in a standalone environment

e. In the **Rollover Policy** section, select **Weekly**, **Daily**, **Hourly**, or **Never** to specify how often you want OracleAS Web Cache to save current log information to *event_log_file.yyyymmdd_hhmm* and write future log information to a new log file with the configured log file name.

Table 15–4 describes additional configuration instructions for **Weekly**, **Daily**, and **Hourly.**

Table 15–4   Configuring Weekly, Daily and Hourly Rollover Policies

Policy	To configure:
Weekly	1. Select a day of the week.
	2. Use the **Time** fields to enter the hour and minutes.
	3. From the **Time Style** list, select either **Local** or **GMT**.
Daily	1. From the **Rollover times - each day** list, select a time. If the time you require does not exist, use the **Time** fields to enter the hour and minutes.
	2. From the **Time Style** list, select either **Local** or **GMT**.
Hourly	1. From the **Rollover times - minutes past the hour** list, select a time. If the time you require does not exist, use the **Minutes past the hour** field to enter the number of minutes, and then click **Add**.
	2. From the **Time Style** list, select either **Local** or **GMT**.

If you have a high-volume site, create a daily or hourly policy.

> **See Also:**   "Rolling Over Event and Access Logs" on page 15-27 for instructions on immediately rolling over log files

    f.   Click **Submit.**

4.   Apply changes and restart OracleAS Web Cache:

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

    b.   In the Cache Operations page, choose **Restart** to restart OracleAS Web Cache.

Event Log Examples

This section contains the following event log examples:

- Example: Event Log with Startup Entries
- Example: Event Log with Unsuccessful Startup Entries
- Example: Event Log with Shutdown Entries
- Example: Event Log with Cache Miss and Cache Hit Entries
- Example: Event Log with an Invalidation Entry
- Example: Event Log with Invalidation Request Errors
- Example: Event Log with ESI Parsing Errors
- Example: Event Log with ESI Processing Errors

Example: Event Log with Startup Entries

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

```
[25/Jul/2005:19:08:52 +0000] [notification 9612] [ecid: -]  OracleAS Web Cache 10g
(10.1.2)
[25/Jul/2005:19:08:52 +0000] [notification 9403] [ecid: -] Maximum number of
file/socket descriptors set to 900.
[25/Jul/2005:19:08:52 +0000] [notification 13002] [ecid: -] Maximum allowed
incoming connections are 700
[25/Jul/2005:16:43:31 -0700] [notification 9607] [ecid: -] The admin server
```

```
started successfully.
[25/Jul/2005:20:44:37 +0000] [notification 12209] [ecid: -] A 1 node cluster
successfully initialized
[25/Jul/2005:19:08:52 +0000] [notification 9608] [ecid: -]  The cache server
process started successfully.
```

Example: Event Log with Unsuccessful Startup Entries

The following shows an event log excerpt with unsuccessful startup events. OracleAS Web Cache is unable to listen on port 7777, because it is already in use. This can occur if OracleAS Web Cache is already running and listening on that port or another application is using that port.

```
[25/Jul/2005:19:12:40 +0000] [notification 9612] [ecid: -]  OracleAS Web Cache 10g
(10.1.2)
[25/Jul/2005:19:12:40 +0000] [notification 9403] [ecid: -] Maximum number of
file/socket descriptors set to 900.
[25/Jul/2005:19:12:40 +0000] [notification 13002] [ecid: -] Maximum allowed
incoming connections are 700
[25/Jul/2005:19:12:40 +0000] [alert 13305] [ecid: -] Failed to assign port 7777:
Address already in use
[25/Jul/2005:19:12:40 +0000] [alert 9707] [ecid: -]  Failed to start the server.
[25/Jul/2005:19:12:40 +0000] [alert 9609] [ecid: -] The server could not
initialize.
[25/Jul/2005:19:12:40 +0000] [notification 9610] [ecid: -]  The server is exiting.
```

Example: Event Log with Shutdown Entries

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

```
[25/Jul/2005:19:09:49 +0000] [notification 9703] [ecid: -] Stop Issued. The
program will shut down after all accepted requests are served, or a timeout
occurs.
[25/Jul/2005:19:09:49 +0000] [notification 9610] [ecid: -]  The server is exiting.
```

Example: Event Log with Cache Miss and Cache Hit Entries

The following shows an event log excerpt containing events for a cache-miss request:

```
[25/Jul/2005:10:30:40 -0700] [req-info] [ecid: 23998683674,0] [client:
10.10.150.35] [host: www.company.com:80] [url: /index.html]
[25/Jul/2005:10:30:40 -0700] [trace 11331] [ecid: 23998683674,0] Request matches
configured site www.company.com:80.
[25/Jul/2005:10:30:40 -0700] [trace 11414] [ecid: 23998683674,0] Initial cache key
is composed: www.company.com:80/index.html
[25/Jul/2005:10:30:40 -0700] [trace 11304] [ecid: 23998683674,0] Cache miss
request.
[25/Jul/2005:10:30:40 -0700] [trace 11224] [ecid: 23998683674,0] Site
www.company.com:80 matches site-to-server mapping www.company.com:80.
[25/Jul/2005:10:30:40 -0700] [trace 11227] [ecid: 23998683674,0] Request is routed
to origin server host-server:7778 using load balancing.
[25/Jul/2005:10:30:40 -0700] [trace 11403] [ecid: 23998683674,0] begin
cacheability decision for following url: www.company.com:80/index.html
[25/Jul/2005:10:30:40 -0700] [trace 11407] [ecid: 23998683674,0] URL matches
caching rule "\.html?$".
[25/Jul/2005:10:30:40 -0700] [trace 11446] [ecid: 23998683674,0] URL which will be
cached is: www.company.com:80/index.html
[25/Jul/2005:10:30:40 -0700] [trace 11415] [ecid: 23998683674,0] Final cache key
is composed: www.company.com:80/index.html
[25/Jul/2005:10:30:40 -0700] [trace 11088] [ecid: 23998683674,0] Following URL is
now in cache: www.company.com:80/index.html
```

The following shows an event log excerpt containing events for a subsequent cache-hit request:

```
[25/Jul/2005:10:32:22 -0700] [req-info] [ecid: 23998788806,0] [client:
10.10.150.35] [host: www.company.com:80] [url: /index.html]
[25/Jul/2005:10:32:22 -0700] [trace 11331] [ecid: 23998788806,0] Request matches
configured site www.company.com:80.
[25/Jul/2005:10:32:22 -0700] [trace 11414] [ecid: 23998788806,0] Initial cache key
is composed: www.company.com:80/index.html
[25/Jul/2005:10:32:22 -0700] [trace 11415] [ecid: 23998788806,0] Final cache key
is composed: www.company.com:80/index.html
[25/Jul/2005:10:32:22 -0700] [trace 11338] [ecid: 23998788806,0] URL is in the
cache
[25/Jul/2005:10:32:22 -0700] [trace 11344] [ecid: 23998788806,0] Returning a
freshly cached object.
```

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 object `cache.htm`.

```
[25/Jul/2005:19:26:46 +0000] [notification 11706][ecid: 21085932167,0] 10 object
(s) matching prefix '/cache/' are invalidated.
[25/Jul/2005:19:26:46 +0000] [notification 11748] [ecid: 21085932167,0]
Invalidation with INFO 'removing 15k object' has returned with status ' SUCCESS';
number of objects invalidated: '10'.
```

Example: Event Log with Invalidation Request Errors

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

```
[25/Jul/2005:19:37:36 +0000] [alert 13109] [ecid: 21086599201,0] XML parsing error
in N/A. Error code 210: LPX-00210: expected '=' instead of 'H'
[25/Jul/2005:19:37:36 +0000] [error 13112] [ecid: 21086599201,0] XML parsing
failed. Error Code: 210
[25/Jul/2005:19:37:36 +0000] [error 11716] [ecid: 21086599201,0]
Invalidation XML message cannot be parsed.
```

Example: Event Log with ESI Parsing Errors

The following shows an event log excerpt with ESI parsing errors that indicate a problem with the `src` attribute and variable syntax:

```
[25/Jul/2005:01:59:48 +0000] [detail] [ecid: 19734744942,0] [client: 127.0.0.1]
[host: www.company.com:80] [url: /cgi-bin/esi-headers.sh?err1.htm]
[25/Jul/2005:01:59:48 +0000] [error 12086] [ecid: 19734744942,0] ESI syntax error.
Unrecognized keyword src is at line 3.
[25/Jul/2005:01:59:48 +0000] [warning 11064] [ecid: 19734744942,0] ESI object
www.company.com:80/cgi-bin/esi-headers.sh?err1.htm parsing error
...
[25/Jul/2005:02:11:57 +0000] [detail] [ecid: 19735486051,0] [client: 127.0.0.1]
[host: www.company.com] [url: /cgi-bin/esi-headers.sh?err1.htm]
[25/Jul/2005:02:11:57 +0000] [error 12095] [ecid: 19735486051,0]  ESI syntax
error. A variable requires a key at line 3.
[25/Jul/2005:02:11:57 +0000] [warning 11064] [ecid: 19735486051,0] ESI object
www.company.com:80/cgi-bin/esi-headers.sh?err1.htm parsing error
```

Example: Event Log with ESI Processing Errors

The following shows an event log excerpt with ESI processing errors that indicate an error fetching an ESI fragment named `a.html` and an unknown fragment for an undefined site:

```
[25/Jul/2005:02:02:37 +0000] [detail] [ecid: 19734914560,0] [client: 127.0.0.1]
[host: www.company.com:80] [url: /cgi-bin/esi-headers.sh?err1.htm]
[25/Jul/2005:02:02:37 +0000] [warning 12005] [ecid: 19734914560,0] HTTP Client
Error 4xx exception in ESI template
www.company.com:80/cgi-bin/esi-headers.sh?err1.htm,
fragment www.company.com:80/cgi-bin/a.html
....
[25/Jul/2005:02:03:38 +0000] [detail] [ecid: 19734980113,0] [client: 127.0.0.1]
[host: -] [url: /cgi-bin/esi-headers.sh?err1.htm]
[25/Jul/2005:02:03:38 +0000] [warning 11295] [ecid: 19734980113,0] OracleAS Web
Cache failed in DNS lookup for host www.company2.com:80.
[25/Jul/2005:02:03:38 +0000] [warning 12003] [ecid: 19734980113,0] No connection
exception in ESI template www.company.com:80/cgi-bin/esi-headers.sh?err1.htm,
fragment www.company2.com:80/a.html
```

Displaying Diagnostic and Event Log Information in the HTML Body or Server Response-Header Field

By default, OracleAS Web Cache adds diagnostics information to the `Server` response-header field. For diagnostics purposes, it can be useful to also display this information as a textual string in the HTML response body of an object. Once enabled, you simply append a string to the URL of the object into the browser to see the diagnostic information string embedded in the response body:

```
Web Cache Debug Info: diagnostic_information
```

You can also select to display event log information, with a verbosity level of TRACE, in the HTML response.

You can additionally configure the diagnostic information to be within HTML comment tags for pages having a `Content-Type: html/html` response-header field. Once enabled, the diagnostic information appears within HTML comment tags:

```
<!-- Web Cache Debug Info: diagnostic_information-->
```

This section contains the following topics:

- Server Response-Header Field
- Configuring Where to Display Diagnostic Information

Server Response-Header Field

For objects sent to browsers, OracleAS Web Cache adds diagnostic information to the `Server` response-header field of the HTTP response message:

```
Server: Oracle Application Server 10g (version) Server_header_from_origin_server
OracleAS-Web-Cache-10g/10.1.2.0.2 (diagnostic_information)
```

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

where `diagnostic_information` has the following format:

```
{ESI_processing_type}{cache_request_type}[;max-age=expiration_time[+removal_
time];age=object_age;]{ecid=request_ID,sequence_number}
```

Table 15–5 on page 15-12 describes the diagnostic fields.

Table 15–5   Control Directives for Server

Control Directive	Description
`ESI_processing_type`	`ESI_processing_type` can be one of the following: ■ `T` specifies that the object is an ESI template ■ `F` specifies that the object 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 object ■ `U` specifies a cache update of a stale object ■ `G` specifies a cache update of an object that was marked for removal but still physically resides in the cache ■ `M` specifies a cache miss ■ `N` specifies a non-cacheable cache miss
`max-age="`*expiration_time*`[+`*removal_time*`]`	Specifies the time, in seconds, to expire the object, and optionally, the time, in seconds, to remove the object from the cache after the expiration time. `max_age` does not appear if the `cache_request_type` is `N`.
`age=`*object_age*	Shows how long, in seconds, the object has been in the cache. `age` does not appear if the object is non-cacheable
`ecid=`*request_ID*`,` *sequence_number*	Specifies the request ID and sequence number specified in `Oracle-ECID` request header: **See Also:** "Oracle-ECID Request-Header Field" on page 15-4

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

```
Server: Oracle Application Server 10g (10.1.2) Oracle-HTTP-Server
OracleAS-Web-Cache-10g/10.1.2.0.2 (TH;max-age=60+30;age=55;ecid=23248098121,0)
```

`(TH;max-age=60+30;age=55;ecid=23248098121,0)` 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 object 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 are 5 seconds to expiration and 35 seconds to removal

■ `ecid=23248098121,0` specifies the request ID and sequence number from the `Oracle-ECID` request header.

To display the `Server` response header in the access logs, you select the `sc(Server)` field. You must configure the `sc(Server)` field as a user-defined access log format.

> **See Also:** "cs(header_name) and sc(header_name) Access Log Fields" on page 15-21 and "Configuring Access Logs" on page 15-22 for further information about creating a user-defined access log format that includes the `sc(Server)` field

Configuring Where to Display Diagnostic Information

To configure diagnostic information in the `Server` response-header field or the HTML response body with Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Diagnostics**.

> **See Also:** "Configuring the Display of Diagnostic Information" in Enterprise Manager Online Help for instructions

To configure diagnostic information in the `Server` response-header field or the HTML response body with OracleAS Web Cache Manager:

1. In the navigator frame, select **Logging and Diagnostics** > **Diagnostics.**

   The Diagnostics page appears.

2. From the Cache-Specific Page Body Diagnostics table, select a cache, and then click **Enable** to display diagnostic information in the HTML response body or **Disable** to disable the display of diagnostic information in the HTML response body.

3. To set diagnostic settings for the HTML response body:

   a. Click **Edit.**

      The Edit Global Page Body Diagnostics Configuration dialog box displays.

   b. In the **URL Flag** field, enter the string to append to the URL of the object.

      By default, the string is set to `+wcdebug`.

   c. In the **Display Event Log Entries for Request** field, select **Yes** to display diagnostic information and TRACE event log entries in the HTML response body, or select **No** to only display diagnostic information.

   d. Click **Submit.**

4. To enable or disable diagnostic settings in the `Server` response header, from the Global Server Header Diagnostics table, click **Enable** or **Disable.**

> **Note:** When a page is compressed, OracleAS Web Cache does not add debug information.

To display diagnostic information in HTML comment tags:

1. Use a text editor to open the `webcache.xml` file.

2. Locate the `<DEBUGINFO>` subelement of `SECURITY`.

   ```
   <DEBUGINFO HEADER="YES" EVENTLOG="NO" HTMLCOMMENT="NO"
   SWITCHSTRING="+wcdebug"/>
   ```

3. Change the value for `HTMLCOMMENT` from `NO` to `YES`.

4. Save `webcache.xml`.

5. Restart OracleAS Web Cache with the following command:

   ```
   opmnctl restartproc ias-component=WebCache
   ```

Evaluating Access Logs

OracleAS Web Cache generates an **access log** that contains information about the HTTP requests sent to OracleAS Web Cache. By default, the access log has a file name

of `access_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 Access Log Files
- Configuring Access Logs
- Analyzing an Access Log File
- Access Log Examples

> **See Also:** *Oracle Application Server Administrator's Guide* for further information about managing log files with Application Server Control Console

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. These fields are based on the standard Extended LogFile Format (XLF). By default, OracleAS Web Cache provides support for the following log formats:

- **Common Log Format (CLF)**

  This format is the default format applied to access logs. This format is appropriate for most configurations. The CLF format provides support for the following fields:

  - `c-ip`
  - `x-log-id`
  - `x-auth-id`
  - `x-clf-date`
  - `x-req-line`
  - `sc-status`
  - `bytes`

- Enhanced CLF (ECLF)

  This format uses many of the CLF fields and includes the `x-ecid` field for tracking the request ID and sequence number specified in `Oracle-ECID` request header:

  - `c-ip`
  - `x-log-id`
  - `x-auth-id`
  - `x-clf-date`
  - `x-req-line`
  - `sc-status`
  - `bytes`
  - `x-ecid`

- Combined Log Format

  This format provides support for the CLF fields with the addition of the `cs(Referer)` and `cs(User-Agent)` fields:

- c-ip

- x-log-id

- x-auth-id

- x-clf-date

- x-req-line

- sc-status

- bytes

- cs(Referer)

- cs(User-Agent)

Select this format when you need to determine what kind of browser is sending the request, and where the browser was visiting before the request was forwarded to OracleAS Web Cache.

■ Enhanced Combined Log Format

This format uses many of the Combined Log Format fields and includes the `x-ecid` field for tracking the ID of the specified in `Oracle-ECID` request header:

- c-ip

- x-log-id

- x-auth-id

- x-clf-date

- x-req-line

- sc-status

- bytes

- cs(Referer)

- cs(User-Agent)

- x-ecid

■ End-User Performance Monitoring Format

This format provides support for the following fields intended for end-user performance monitoring features:

- x-req-type

- x-date-start

- x-time-start

- c-ip

- s-ip

- x-auth-id

- cs(Host)

- cs-method

- cs-uri

- x-protocol

- sc-status
- bytes
- cs-bytes
- x-cache
- time-taken
- r-time-taken
- x-time-delay
- x-os-timeout
- x-ecid
- x-cookie(ORACLE_SMP_CHRONOS_ST)
- x-cookie(ORACLE_SMP_CHRONOS_LT)
- x-cookie(ORACLE_SMP_CHRONOS_GL)
- x-glcookie-set
- cs(Referer)
- cs(User-Agent)
- x-esi-info
- x-conn-abrt
- sc(Content-Type)

If the default formats are not suitable for your environment, you can create custom log formats by specifying the fields that you require. Table 15–6 describes the supported fields. Fields prefixed with x or r are proprietary to OracleAS Web Cache.

Table 15–6   Access Log Fields

Field	Description
bytes	Content length of the request
c-ip	IP address of the client
cached	Integer that specifies cache status. Cache status is reported as one of the following: ■ 0 specifies a cache miss. Equivalent to M, U, G, and N output of x-cache field. ■ 1 specifies a cache hit of a stale object. Equivalent to S output of x-cache field. ■ 2 specifies a cache hit. Equivalent to H output of x-cache field.
cs(*header_name*)	HTTP request header sent from the client **See Also:** "cs(header_name) and sc(header_name) Access Log Fields" on page 15-21
cs-bytes	Bytes received from the client
cs-method	Client-to-OracleAS Web Cache HTTP request method
cs-uri	Client-to-OracleAS Web Cache URI
cs-uri-query	Client-to-OracleAS Web Cache query portion of URI, omitting the stem
cs-uri_stem	Client-to-OracleAS Web Cache stem portion of URI, omitting the query

Table 15–6 (Cont.) Access Log Fields

Field	Description
date	Date the transaction completed, in the following format: *dd/Mon/yyyy*
r-ip	IP address and port number of origin server. For a cache cluster, this field displays the IP and port number of a peer cache in the cache cluster. The information is displayed in the following format: *IP_address:port*
r-time-taken	Time, in seconds (including microseconds), that OracleAS Web Cache spent communicating with the origin server or peer cache. The time is the duration between the following two points of time: ■ The time immediately before OracleAS Web Cache sent the first byte of the request to the origin server or peer cache. ■ The time immediately after receiving the last byte of the response from the origin server or peer cache. This field is particularly helpful in providing time information for end-user performance monitoring.
s-ip	IP address of OracleAS Web Cache computer
sc(*header_name*)	HTTP response header sent from OracleAS Web Cache to the client **See Also:** "cs(header_name) and sc(header_name) Access Log Fields" on page 15-21
sc-status	OracleAS Web Cache-to-client 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 to complete the request ■ 4*xx* range messages indicate a client error ■ 5*xx* range messages indicate a OracleAS Web Cache error **See Also**: http://www.ietf.org/rfc/rfc2616.txt for further information about HTTP status codes
time	Time at which the response from OracleAS Web Cache completed. The time is displayed in the following format: *hh:mm:ss*
time-taken	Amount of time taken, in seconds (including microseconds), for the transaction to complete
x-auth-id	User name of a basic HTTP authentication request
x-cache	Cache status. Cache status is reported as one of the following: ■ H specifies a cache hit ■ S specifies a cache hit of a stale object ■ U specifies a cache update of a stale object ■ G specifies a cache update of an object that was marked for removal but still physically resides in the cache ■ M specifies a cacheable cache miss ■ N specifies a non-cacheable cache miss

Table 15–6   (Cont.) Access Log Fields

Field	Description
x-cache-detail	Diagnostic information, in the following format:
	{*ESI_processing_type*}{*cache_request_type*} [;max-age=*expiration_time*[+*removal_time*];age=*object_age*]
	ESI_processing_type can be one of the following:
	- T specifies that the object is an ESI template
	- F specifies that the object is an ESI fragment
	- Empty specifies that the response does not require ESI processing
	cache_request_type can be one of the following:
	- H specifies a cache hit
	- S specifies a cache hit of a stale object
	- U specifies a cache update of a stale object
	- G specifies a cache update of an object that was marked for removal but still physically resides in the cache
	- M specifies a cacheable cache miss
	- N specifies a non-cacheable cache miss
	max_age specifies the time, in seconds, to expire the object, and optionally, the time, in seconds, to remove the object from the cache after the expiration time. max_age does not appear if the *cache_request_type* is N.
	age shows how long, in seconds, the object has been in the cache. age does not appear if the object is non-cacheable.
	Example: H;max-age=60+30;age=50
	- H means that this request resulted in cache hit
	- max-age=60+30 means that the object 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=50 means that 50 seconds have passed since population of the cache, meaning there is 10 seconds to expiration and 40 seconds to removal
x-cache-key	Cache key value, in the following format:
	"*cache_key*"
x-clf-date	Date that the response from OracleAS Web Cache completed, in the following format:
	dd/Mon/yyyy:hh:mm:ss [+GMT]
x-cluster	Single character that specifies the status of a cache cluster. The character is reported as one of the following:
	- T specifies a request to a cache cluster member
	- F specifies a request from a cache cluster member
	- O specifies a request for owned content
	- D specifies a request for on-demand content
x-cookie(*cookie_name*)	Cookie value from client browser request.

Table 15–6   (Cont.) Access Log Fields

Field	Description
x-conn-abrt	Single character that specifies the whether or not a connection was terminated before a response was completed. This field is intended for end-user performance monitoring. ■ C specifies that the connection was terminated by the client before OracleAS Web Cache could complete a response. ■ O specifies that the connection was terminated by the origin server before it could complete a response to OracleAS Web Cache. ■ N specifies the response was completed without the connection being terminated.
x-date-start	Date before OracleAS Web Cache received the first byte of the request, in the following format: *yyyy-mm-dd*
x-date-end	Date when OracleAS Web Cache sent the last byte of the response, in the following format: *yyyy-mm-dd*
x-ecid	ID of the specified in Oracle-ECID request header, in the following format: "*request_ID, sequence_number*" **See Also:** "Oracle-ECID Request-Header Field" on page 15-4
x-esi-info	ESI fragment log message from the log element of <esi:environment> or <esi:include> tags. It uses the following format: "*ESI_log_message*" The log message only displays for requested ESI fragments in the *access_log_file*.fragment file. When a request ESI fragment is not configured with the log element, this field displays as a hyphen (-)
x-glcookie-set	Boolean character that specifies whether or not OracleAS Web Cache created the ORACLE_SMP_CHRONOS_GL cookie and sent as a response to the client browser a Set-Cookie:ORACLE_SMP_CHRONOS_GL response header field. This field is intended for end-user performance monitoring to track transactions. ■ Y specifies that OracleAS Web Cache set the ORACLE_SMP_CHRONOS_GL cookie. Y also marks the beginning of a transaction for the client. All subsequent traffic from the browser send a Cookie request-header field set with the ORACLE_SMP_CHRONOS_GL cookie received in the OracleAS Web Cache response. ■ N specifies that OracleAS Web Cache did not create the cookie. This result can occur because the cookie is already set.
x-log-id	Login user name of the client. OracleAS Web Cache is unable to obtain the value for this field. Therefore, OracleAS Web Cache displays a hyphen (-) in the output when this field is set.
x-os-name	Origin server or cache cluster member that OracleAS Web Cache is forwarding the request, in the following format: *host:port*
x-os-timeout	Single character that specifies if the origin server timed out on a request. The character is reported as one of the following: ■ 0 specifies that the origin server did not timeout ■ 1 specifies that the origin server did timeout. An output of 1 can indicate a problem with the origin server itself.
x-protocol	Protocol and version from client request, in the following format: *protocol/version*

Table 15–6 (Cont.) Access Log Fields

Field	Description
x-req-line	Request line, in the following format: "*HTTP_request_method URI protocol/version*" **Example:** "GET /cache.htm HTTP/1.1"
x-req-type	Request type. Request type is reported as one of the following: ■ B specifies that the request is from the browser ■ C specifies that the request is from another cache cluster member ■ H specifies that the request is from another cache cluster or an OracleAS Web Cache that is not a member of the current cache cluster ■ F specifies that the request is for an ESI fragment
x-time-delay	Time, in seconds (including microseconds), that OracleAS Web Cache spent communicating with the origin server or peer cache. The time is the duration between the following two points of time: ■ The time immediately before OracleAS Web Cache received the first byte of the request ■ The time immediately before OracleAS Web Cache sent the first byte of the request to the origin server or peer cache. This field is particularly helpful in providing time information for End-User Performance Monitoring.
x-time-end	Time that OracleAS Web Cache sent the last byte of the response, in the following format: *hh:mm:ss:ssssss*
x-time-handshake	The difference between the times the client initiates a new connection and the time at which OracleAS Web Cache receives the first byte of the HTTP request. **Note:** Select this field only if instructed by Oracle Support Services.
x-time-reqrecvlatency	The difference between the times OracleAS Web Cache receives the first and last byte of the HTTP request. This field indicates the time in reading the browser requests. **Note:** Select this field only if instructed by Oracle Support Services.
x-time-reqsendlatency	The difference between the times OracleAS Web Cache sends the first and last byte of the HTTP request to the origin server. This field indicates the time taken in sending the request to the origin server. **Note:** Select this field only if instructed by Oracle Support Services.
x-time-resprecvlatency	The difference between the times OracleAS Web Cache receives the first and last byte of the HTTP response from the origin server. This field indicates the time taken in receiving the response from the origin server. **Note:** Select this field only if instructed by Oracle Support Services.
x-time-respsendlatency	The difference between the times OracleAS Web Cache sends the first and last byte of the HTTP response to the browser. This field indicates the time taken in sending the response to the client. **Note:** Select this field only if instructed by Oracle Support Services.

Table 15–6   (Cont.) Access Log Fields

Field	Description
x-time-reqblocked	The difference between when a request was blocked and unblocked due to a cache update. If a request has already been sent to the origin server by OracleAS Web Cache to update an existing object, OracleAS Web Cache blocks all subsequent requests.
	Note: Select this field only if instructed by Oracle Support.
x-time-reqqueued	The difference between when a request is queued and dequeued for the origin server. This field indicates the time a request spends in OracleAS Web Cache backend queue for an origin server (due to the maximum origin server capacity being reached) before the request is sent to the origin server for processing.
	Note: Select this field only if instructed by Oracle Support.
x-time-start	Time before OracleAS Web Cache received the first byte of the request, in the following format:
	hh:mm:ss:ssssss

cs(*header_name*) and sc(*header_name*) Access Log Fields

Table 15–7 lists examples of HTTP/1.1 headers that can be used for the cs(*header_name*) and sc(*header_name*) fields. This table lists only some of the possible headers. It is not an exhaustive list.

Table 15–7   Examples of HTTP/1.1 Header Fields

cs(*header_name*) Field	sc(*header_name*) 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	
Pragma	
Range	
TE	
User-Agent	
Via	

Table 15–8 lists examples of cookie-related headers that can be used for the cs(*header_name*) and sc(*header_name*) fields.

Table 15–8   Supported Cookie-Related Header Fields

cs(*header_name*) Field	sc(*header_name*) Field
Cookie	Set-Cookie

Table 15–9 lists examples of OracleAS Web Cache headers that can be used for the cs(*header_name*) and sc(*header_name*) fields.

Table 15–9   Supported OracleAS Web Cache Header Fields

cs(*header_name*) Field	sc(*header_name*) Field
Surrogate-Capability	Surrogate-Control

Configuring Access Logs

To configure the access log settings with Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Properties** > **Logging**.

> **See Also:**   "Configuring Access Logs" in Enterprise Manager Online Help for instructions

To establish access log configuration settings with OracleAS Web Cache Manager:

1. In the navigator frame, select **Logging and Diagnostics** > **Access Logs**.

   The Access Logs page appears.

2. Specify cache-specific access log settings:

   a. From the Cache-Specific Access Log Configuration table, select a cache, and then click **Edit Selected**.

      The Edit Cache-Specific Access Log Configuration dialog box appears.

   b. In the **Directory** field, enter the directory in which to write access logs.

      The default is $ORACLE_HOME/webcache/logs on UNIX and *ORACLE_HOME*\webcache\logs on Windows.

   c. In the **Enabled** field, select **Yes** to enable logging, or **No** to disable logging.

   d. In the **Buffering** field, select **Enabled** to enable buffered logging or **Disabled** to disable buffered logging.

      With buffered logging, OracleAS Web Cache writes to the access log after the buffer is full. The buffer size is set to 2048 bytes. When the limit is reached, OracleAS Web Cache writes buffered events to the access log file.

      Oracle recommends disabling buffering when you need to see the access log results immediately.

   e. If buffering is enabled, in the **Flush Interval** field, enter the maximum threshold, in seconds, when contents of the buffer are written to the access log file.

      The default is 10 seconds. When the maximum threshold is reached, OracleAS Web Cache writes buffered information to the access log file. Even if the buffer is not full, the access log is updated at least every 10 seconds. Oracle

recommends not changing the default, unless you need to lower the interval to see results more frequently.

   f. Click **Submit.**

3. Specify site-specific log settings:

   a. From the Site-Specific Access Log Configuration table, click **Add.**

     The Edit/Add Site Specific Access Log Configuration dialog box appears.

   b. From the **For Site** list, select the Web site for which to specify access log settings.

   c. In the **File Name** field, enter a name for the access log file.

     The default file name is `access_log`.

   d. In the **Enabled** field, select **Yes** to enable logging for the site or **No** to disable logging for the site.

     Site-specific logging only takes effect if logging is enabled for the cache. If you select **Yes**, ensure that **Yes** is also selected for the cache in Step 2c.

   e. In the **ESI Fragment Requests** field, select **Log** to log the ESI fragment log messages from the `log` element of `<esi:environment>` or `<esi:include>` in the `access_log_file`.`fragment` file.

     If the `x-esi-info` field is selected, select **Log** to log the events to the `access_log_file`.`fragment` file. The `x-esi-info` field is automatically selected if the **Format Style** is End-User Performance Monitoring Format. If the `x-esi-info` field is not selected, select **Don't Log.**

   f. From the **Format Style** list, select an access log format.

     See Also: "Format of the Access Log Files" on page 15-14 for a description of the default formats

   g. From the **Rollover Policy** list, select a rollover policy to specify how often you want to change the frequency at which OracleAS Web Cache saves current log information to `access_log_file`.`yyyymmdd_hhmm` and writes future log information to a new log file with the configured log file name.

     For high-volume sites, select a policy with a high frequency.

   h. Click **Submit.**

4. If the default formats are not suitable for your environment, create a log format that is:

   a. From the User-Defined Log Formats table, click **Add.**

     The Edit/Add User-Defined Access Log Format dialog box displays.

   b. In the **Format Name** field, enter a unique name for the format.

   c. From the **Separator** list, select the separator to use for separating access log fields.

   d. In **Print XLF Directive** field, select **Yes** to include XLF directive information at the top of the access log or **No** to not include directive information in the access log.

     Directive information typically consists of version, date, and field information. For example:

```
#Version: 1.0
#Date: 12-Jul-2005 00:00:00
#Fields: c-ip x-auth-id x-clf-date cs(Host x-req-line sc-status bytes
```

> **See Also:** http://www.w3.org/TR/WD-logfile.html for
> further information about XLF directives

    e. In the **XLF Fields** section, select an access log field name from the **Field name** list.

> **See Also:** Table 15–6 on page 15-16 for a listing of the supported access logs fields

    f. If you selected field `cs(header_name)`, `sc(header_name)`, or `x-cookie(cookie_name)`, then enter the header or cookie name in the **Header/Cookie name** field.

> **See Also:** Table 15–7 on page 15-21, Table 15–8 on page 15-22, and
> Table 15–9 on page 15-22 for a description of the headers allowed for
> `cs(header_name)` and `sc(header_name)`

    g. Click **Add.**

    h. Perform Steps e and f for each format you want in the access log, and then use the **Move Up** and **Move Down** buttons to order the fields. The order in which fields are entered determines the order in which the fields are logged.

    i. Click **Submit.**

5. Optionally, modify or create rollover policies:

    a. Select an existing policy and click **Edit Selected** to modify and existing rollover policy, or click **Add** to create a new policy.

        The Edit/Add Access Log Rollover Policy dialog box appears.

    b. In the **Rollover Policy Name** field, enter a unique name for the rollover policy.

    c. In **Rollover Policy** section, select **Weekly**, **Daily**, **Hourly**, or **Never** to specify how often you want OracleAS Web Cache to save current log information to *access_log_file.yyyymmdd_hhmm* and write future log information to a new log file with the configured log file name.

        Table 15–10 describes additional configuration instructions for **Weekly**, **Daily**, and **Hourly.**

Table 15–10   Configuring Weekly, Daily and Hourly Rollover Policies

Policy	To configure:
Weekly	**1.** Select a day of the week.
	2. Use the **Time** fields to enter the hour and minutes.
	3. From the **Time Style** list, select either **Local** or **GMT (Greenwich Mean Time)**.
Daily	**1.** From the **Rollover times - each day** list, select a time. If the time you require does not exist, use the **Time** fields to enter the hour and minutes, and then click **Add**.
	2. From the **Time Style** list, select either **Local** or **GMT**.

Table 15–10 (Cont.) Configuring Weekly, Daily and Hourly Rollover Policies

Policy	To configure:
Hourly	1. From the **Rollover times - minutes past the hour** list, select a time. If the time you require does not exist, use the **Minutes past the hour** field to enter the number of minutes, and then click **Add**.
	2. From the **Time Style** list, select either **Local** or **GMT**.

If you have a high-volume site, create a daily or hourly policy.

See Also: "Rolling Over Event and Access Logs" on page 15-27 for instructions on immediately rolling over log files

   d. Click **Submit.**

6. Apply changes and restart OracleAS Web Cache:

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

   b. In the Cache Operations page, choose **Restart.**

Logging Cache Cluster Peer Requests

By default, peer requests between two members of a cache cluster are not logged in the access log. Only client requests to the cluster are logged. Peer request logging can be enabled for individual cache cluster members by adding the `ACCESSLOGIGNOREPEERREQUEST` attribute to the `MISCELLANEOUS` element in the `internal.xml` configuration file.

The valid values for this attribute are: `YES` and `NO`.

The following example shows the `MISCELLANEOUS` element with peer-to-peer logging enabled:

```
<MISCELLANEOUS ACCESSLOGIGNOREPEERREQUEST="NO"/>
```

Analyzing an Access Log File

The following code shows an excerpt of an access log file:

```
10.10.150.35 - - [25/Jul/2005:10:27:42 -0500] "GET /~user/personal.htm HTTP/1.1"
200 2438
10.10.150.35 - - [25/Jul/2005:10:27:54 -0500] "GET
/~user/personal.htm?UserName=Bob HTTP/1.1" 200 2438
10.10.150.35 - - [25/Jul/2005:10:47:30 -0500] "GET /~user/count.sh HTTP/1.1" 403
289
10.10.150.35 - - [25/Jul/2005:10:47:34 -0500] "GET /~user/sbin/count.sh HTTP/1.1"
200 321
```

In the first line of the output, the fields have the following meaning:

- `10.10.150.35` is the browser's IP address (`c-ip`)

- `[25/Jul/2005:10:27:42 -0500]` is the date (`[x-clf-date]`)

- `"GET /~user/personal.htm HTTP/1.1"` is the request line (`"x-req-line"`)

- `200` is the HTTP status code (`sc-status`)

- `2438` is the size of the object sent (`bytes`)

Access Log Examples

This section contains the following access log examples:

- Example: Access Log with Reload Entries
- Example: Access Log with Status Code 404 Entry
- Example: Access Log in Combined Format
- Example: Access Log with Site Information
- Example: Access Log with ESI Diagnostic Information
- Example: Access Log with ESI Log Information

Except where noted otherwise, the access log examples use the CLF format:

```
c-ip x-log-id x-auth-id x-clf-date x-req-line sc-status bytes
```

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:

```
10.10.150.35 - - [25/Jul/2005:11:04:24 -0500] "GET /cache.htm HTTP/1.1" 200 250
10.10.150.35 - - [25/Jul/2005:11:04:26 -0500] "GET /cache.htm HTTP/1.1" 200 250
10.10.150.35 - - [25/Jul/2005:11:29:24 -0500] "GET /cache.htm HTTP/1.1" 304 0
10.10.150.35 - - [25/Jul/2005:11:29:25 -0500] "GET /cache.htm HTTP/1.1" 304 0
10.10.150.35 - - [25/Jul/2005:11:29:30 -0500] "GET /cache.htm HTTP/1.1" 200 250
10.10.150.35 - - [25/Jul/2005:11:29:35 -0500] "GET /cache.htm HTTP/1.1" 200 250
```

The third and forth entries return an HTTP status code of 304, indicating that object 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 OracleAS Web Cache cannot find any objects matching the requested URL /ows-img/chalk.jpg. This error is indicated by HTTP status code 404.

```
10.10.150.35 - - [25/Jul/2005:10:49:44 -0500] "GET /pls/coe/find_via_post
HTTP/1.1" 200 1119
10.10.150.35 - - [25/Jul/2005:10:49:44 -0500] "GET /ows-img/chalk.jpg HTTP/1.1"
404 284
```

Example: Access Log in Combined Format

The following shows an access log excerpt in which the combined format is specified:

```
c-ip x-log-id x-auth-id x-clf-date x-req-line sc-status bytes cs(Referer)
 cs(User-Agent)

10.10.150.35 - - [25/Jul/2005:20:09:47 +0000] "GET /manual/sections.html HTTP/1.1"
200 -1 "http://www.company.com:80/manual/mod/directive-dict.html#Syntax"
"Mozilla/4.78 [ja] (Win98; U)"
10.10.150.35 - - [25/Jul/2005:20:09:50 +0000] "GET /manual/mod/core.html HTTP/1.1"
200 -1 "http://www.company.com:80/manual/sections.html" "Mozilla/4.78 [ja] (Win98;
U)"
10.10.150.35 - - [25/Jul/2005:20:10:06 +0000] "GET / HTTP/1.1" 200 -1 -
"Mozilla/4.78 [ja] (Win98; U)"
10.10.150.35 - - [25/Jul/2005:20:10:14 +0000] "GET /manual/LICENSE HTTP/1.1" 200
-1 "http://www.company.com:80/manual/index.html" "Mozilla/4.78 [ja] (Win98; U)"
```

Example: Access Log with Site Information

The following shows an access log excerpt in which the following fields are specified:

```
c-ip x-auth-id x-clf-date cs(Host) x-req-line sc-status bytes
```

cs(Host) displays the output of Host request-header field, which specifies the site information. In this example, requests are sent to OracleAS Web Cache for site www.company.com:80.

```
10.10.150.35 - [25/Jul/2005:20:05:51 +0000] "www.company.com:80" "GET / HTTP/1.1"
200 -1
10.10.150.35 - [25/Jul/2005:20:05:56 +0000] "www.company.com:80" "GET
/manual/index.html HTTP/1.1" 200 -1
10.10.150.35 - [25/Jul/2005:20:05:59 +0000] "www.company.com:80" "GET
/manual/upgrading_to_1_3.html HTTP/1.1" 200 -1
10.10.150.35 - [25/Jul/2005:20:06:02 +0000] "www.company2.com:80" "GET
/manual/mod/mod_dir.html HTTP/1.1" 200 -1
10.10.150.35 - [25/Jul/2005:20:06:05 +0000] "www.company2.com:80" "GET
/manual/mod/directive-dict.html HTTP/1.1" 200 -1
```

Example: Access Log with ESI Diagnostic Information

The following shows an access log excerpt in which the following fields are specified:

```
c-ip x-clf-date x-req-line sc-status bytes x-cache-detail
```

x-cache-detail displays diagnostic information. In the following example:

- T means that this request is for an ESI template

- H means that this request resulted in cache hit

- max-age=10+15 means that the object is to expire in 10 seconds from population and to be removed from the cache 15 seconds from the expiration. This provides a total of 25 seconds from population.

- age=0 means that 0 seconds have passed since population of the cache, meaning there is 10 seconds to expiration and 15 seconds to removal

```
[25/Jul/2005:02:35:37 +0000] "GET /cgi-bin/esi-headers.sh?err1.htm HTTP/1.0" 200
42 TM;max-age=10+15;age=0
```

Example: Access Log with ESI Log Information

The following shows an access log excerpt in which the following fields are specified:

```
c-ip x-clf-date x-req-line sc-status bytes x-esi-info
```

x-esi-info displays log information from the log element of <esi:environment> or <esi:include> tags.

```
[25/Jul/2005:03:03:35 +0000] "GET /b.html HTTP/1.0" 200 4 "This is a sample
fragment."
```

Rolling Over Event and Access Logs

In addition to configuring event and access log rollover frequency, you can also use OracleAS Web Cache Manager to immediately roll over event and access logs. During the rollover process, OracleAS Web Cache saves current information to log_file.*yyyymmdd_hhmm* and writes future log information to a new log file with the configured log file name.

To immediately roll over log files:

To immediately roll over log files with Application Server Control Console, navigate to **Web Cache Home** page > **Administration** tab > **Operations** > **Rollover Log Files**.

> **See Also:** "Rolling Over Log Files" in Enterprise Manager Online Help for instructions

To immediately roll over log files with OracleAS Web Cache Manager:

1. In the navigation pane, select **Operations** > **On-Demand Log File Rollover**.

   The On-Demand Log File Rollover page appears.

2. To roll over event log files:

   a. From the **Event Logs** table, select an individual cache, or click **Select All** to select all the caches.

   b. Click **Submit.**

3. To roll over access log files:

   a. From the **Access Logs** table, select an access log for a configured site, or click **Select All** to select all the access logs.

   b. Click **Submit.**

Part III

Reference

Part III provides reference material for this guide.

This part contains these chapters:

16

Edge Side Includes (ESI) Language Tags

This chapter describes the **Edge Side Includes (ESI)** tags provided for content assembly of dynamic fragments.

This chapter contains these topics:

- Overview of ESI
- ESI Tag Descriptions

Overview of ESI

ESI is an open specification co-authored by Oracle. Its purpose is to develop a uniform programming model to assemble dynamic pages in a dynamic content cache deployed as a surrogate or proxy between clients and origin servers.

ESI is an XML-like markup language that enables dynamic content assembly of fragments by OracleAS 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 caching rules to the template page and HTML fragments. By enabling page assembly in OracleAS Web Cache rather than in the origin server, you can increase cache hit rates and improve performance.

> **See Also:**
>
> - "Configuring Rules for Content Assembly and Partial Page Caching" on page 12-33
> - "Configuring Expiration Policies" on page 12-14
> - `http://www.esi.org` for the ESI language release 1.0 specification

The following topics provide an overview of ESI usage:

- About the Surrogate-Control Response Header for Supporting ESI Language Elements
- About the Surrogate-Capability Request Header for Cached Objects
- Syntax Rules
- Nesting Elements
- Variable Expressions
- Exceptions and Errors

About the Surrogate-Control Response Header for Supporting ESI Language Elements

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

OracleAS Web Cache supports the ESI language tags, elements, and attributes listed in Table 16–1. The rightmost column specifies, for each ESI tag, attribute, or element, all the feature sets that support it. For example, the `<esi:invalidate>` tag is only supported by the `"ESI-INV/1.0"` feature set. To enable the correct processing in OracleAS Web Cache, specify all the feature sets that an ESI template uses in the `content` control directive of the `Surrogate-Control` response header. However, you do need to specify features sets used within an ESI fragment directly or indirectly included in the template. For example, if an ESI template uses an `<esi:invalidate>` and an `<esi:environment>` tag with an `<esi:log>` element, the `content` control directive must include `"ESI-INV/1.0"` and `"ORAESI/9.0.4"`, as follows:

```
Surrogate-Control: content="ESI-INV/1.0 ORAESI/9.0.4"
```

> **See Also:** "Using the Surrogate-Control Response Header as an Alternative to Caching Rules" on page 12-30 for further information about configuring the `Surrogate-Control` response header

Table 16–1   ESI Language Features

ESI Language Feature	See Also	content="*value*" Control Directive in Surrogate-Control Response Header Supporting Feature
`<esi:choose>` | `<esi:when>` | `<esi:otherwise>` tags	"ESI choose | when | otherwise Tags" on page 16-12	`"ORAESI/9.0.4"` `"ORAESI/9.0.2"` `"ESI/1.0"`
`<esi:comment>` tag	"ESI comment Tag" on page 16-15	`"ORAESI/9.0.4"` `"ORAESI/9.0.2"` `"ESI/1.0"`
`<esi:environment>` tag	"ESI environment Tag" on page 16-16	`"ORAESI/9.0.4"` `"ORAESI/9.0.2"`
`<esi:include>` tag	"ESI include Tag" on page 16-19	`"ORAESI/9.0.4"` `"ORAESI/9.0.2"` `"ESI/1.0"`
`<esi:environment>` tag and `<esi:include>` tag attributes and elements		
`alt` attribute	"ESI include Tag" on page 16-19	`"ORAESI/9.0.4"` `"ESI/1.0"`
`max-age` attribute	"ESI include Tag" on page 16-19 "ESI environment Tag" on page 16-16	`"ORAESI/9.0.4"` `"ORAESI/9.0.2"`
`onerror` attribute	"ESI include Tag" on page 16-19	`"ORAESI/9.0.4"` `"ORAESI/9.0.2"` `"ESI/1.0"`
`src` attribute	"ESI include Tag" on page 16-19	`"ORAESI/9.0.4"` `"ORAESI/9.0.2"` `"ESI/1.0"`

Table 16–1 (Cont.) ESI Language Features

ESI Language Feature	See Also	content="*value*" Control Directive in Surrogate-Control Response Header Supporting Feature
`timeout` attribute	"ESI include Tag" on page 16-19	`"ORAESI/9.0.4"`
	"ESI environment Tag" on page 16-16	`"ORAESI/9.0.2"`
`<esi:log>` element	"ESI include Tag" on page 16-19	`"ORAESI/9.0.4"`
	"ESI environment Tag" on page 16-16	
`<esi:request_header>` element	"ESI include Tag" on page 16-19	`"ORAESI/9.0.4"`
	"ESI environment Tag" on page 16-16	`"ORAESI/9.0.2"`
`<esi:request_body>` element	"ESI include Tag" on page 16-19	`"ORAESI/9.0.4"`
	"ESI environment Tag" on page 16-16	`"ORAESI/9.0.2"`
`<esi:inline>` tag	"ESI inline Tag" on page 16-23	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI-Inline/1.0"`
		`"ESI/1.0"`
`name` attribute	"ESI inline Tag" on page 16-23	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI-Inline/1.0"`
		`"ESI/1.0"`
`fetchable` attribute	"ESI inline Tag" on page 16-23	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI-Inline/1.0"`
		`"ESI/1.0"`
`max-age` attribute	"ESI inline Tag" on page 16-23	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
`timeout` attribute	"ESI include Tag" on page 16-19	`"ORAESI/9.0.4"`
	"ESI environment Tag" on page 16-16	`"ORAESI/9.0.2"`
`<esi:invalidate>` tag	"ESI invalidate Tag" on page 16-25	`"ESI-INV/1.0"`
`<esi:remove>` tag	"ESI remove Tag" on page 16-27	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI/1.0"`
`<esi:try>` | `<esi:attempt>` | `<esi:accept>` tags	"ESI try | attempt | except Tags" on page 16-28	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI/1.0"`
`<esi:vars>` tag	"ESI vars Tag" on page 16-32	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI/1.0"`
`<!--esi...-->` tag	"ESI <!--esi-->Tag" on page 16-34	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI/1.0"`

> **See Also:** http://www.esi.org/spec.html for the *ESI Language Specification 1.0* and the *Edge Architecture Specification*

About the Surrogate-Capability Request Header for Cached Objects

For each requested object from the cache, OracleAS Web Cache appends a `Surrogate-Capability` request-header field to an object's HTTP request message. The `Surrogate-Capability` request-header serves the following purposes:

- Enables applications to detect OracleAS Web Cache

- Identifies the types of ESI operations that OracleAS Web Cache can perform

The `Surrogate-Capability` request-header enables OracleAS Web Cache to identify the operations it is capable of performing to origin servers acting as caches. 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/9.0.4"` to process ESI tags with Oracle-proprietary additions for content assembly and partial page caching. `"ORAESI/9.0.4"` supports all the ESI tags provided by OracleAS Web Cache in 10*g* (9.0.4) and later releases.

- `"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 OracleAS Web Cache in Release 2 (9.0.2 and 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

- `"ESI-INV/1.0"` to process `<esi:invalidate>` tags

- `"webcache/1.0"` to process the `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` tags for personalized attributes

The values `"ORAESI/9.0.2"`, `"ESI/1.0"`, and `"ESI-Inline/1.0"` are subsets of `"ORAESI/9.0.4"`. For this release, you specify only `"ORAESI/9.0.4"` for ESI assembly, `"ESI-INV/1.0"` for inline invalidation, or `"webcache/1.0"` for personalized attributes.

> **See Also:** Table 16–2 on page 16-7 for further information about the ESI tags supported for each "`operation_value`"

Syntax Rules

ESI elements and attributes adhere to XML syntax but can be embedded in other objects, such as HTML or XML objects. When OracleAS 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 Example 16–1, an ESI tag can contain nested ESI elements and other HTML markup.

Example 16–1   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

ESI supports the HTTP request variables and environment variables used with the `<esi:environment>` tag.

This section contains the following topics:

- Variable Usage

- Variable Default Values

- HTTP Request Variables

> **See Also:** "ESI environment Tag" on page 16-16 for instructions on including custom variables

Variable Usage

To refer to a variable, prefix it with a dollar sign and surround the variable name with parentheses:

```
$(VARIABLE_NAME)
```

For example:

```
$(HTTP_HOST)
```

Variables 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, the environment variable returns the whole content of the environment fragment. Oracle 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:environment src="/getlogindata" name="logindata"/>
<esi:include src="/login/$(logindata{account})"/">
<esi:choose>
  <esi:when test="$(logindata) != null">
    <esi:include src="/login/$(logindata{account})"/>
 </esi:when>
 <esi:otherwise>
   <esi:include src="/login/guest.html"/>
 <esi:otherwise>
</esi:choose>
```

Variable Default Values

You can use the logical OR (|) operator to specify a default value in the following form:

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

A variable takes the default value only when the variable or its key does not exist. If it evaluates to an empty string, the empty string is returned, not the default value.

The following example results in OracleAS 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, 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.

HTTP Request Variables

Table 16–2 on page 16-7 lists the HTTP request variables supported by ESI. Note the following:

- 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.

- Variables with a substructure type of List or Dictionary are accessed by a key.

- Variables identified with a substructure type of Dictionary make access to strings available through their appropriate keys.

- Dictionary keys are case sensitive.

- Variables identified with a substructure type of List return a Boolean value depending on whether the requested value is present.

Table 16–2   HTTP Request Variables Supported by ESI

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 the language specified in the HTTP request header	Variable Setting:
				`$(HTTP_LANGUAGE{en-gb})`
				HTTP Request Header Contains:
				`Accept_Language:en-gb`
				Result: Returns `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, 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 `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
				`http://www.company.com: 80`

`$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
				`http://www.company.com: 80`

Table 16–2   (Cont.) HTTP Request Variables Supported by ESI

Variable Name	HTTP Header Field	Substructure Type/Variable Type	Description	Example

`$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:
				`$(HTTP_USER_ AGENT{browser})`
				HTTP Request Header Contains:
				`User-Agent:Mozilla/4.0 (compatible, MSIE 5.5, Windows)`
				Result: Returns `MSIE`
				`$(HTTP_USER_ AGENT{version})`
				Result: Returns `4.0.`
				`$(HTTP_USER_AGENT{os})`
				Result: Returns `Windows`

`$(QUERY_ STRING{parameter})`	Not Applicable	Dictionary/ String	Given a parameter name in a query string, returns the value of the parameter without URL encoding. The query string can be in an URL or a request body.
			See Also: http://www.rfc-editor.org/ for further information about URL encoding.
				Variable Setting:
				`$(QUERY_STRING{CEO})`
				Query Request Contains:
				`CEO=Jane%20Doe&CFO=John %20Doe`
				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)`
				Query Request Contains:
				`CEO=Jane%20Doe&CFO=John %20Doe`
				Result: Returns the entire query string encoded:
				`CEO=Jane%20Doe&CFO=John %20Doe`

Table 16–2   (Cont.) HTTP Request Variables Supported by ESI

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{fullname})` Query Request Contains: `fullname=Jane%20Doe` Result: Returns the value of `fullname` encoded: `Jane%20Doe`
`$(QUERY_STRING_ ENCODED)`	Not Applicable	Not Applicable/ String	The same as `$(QUERY_STRING)`	Variable Setting: `$(QUERY_STRING_ENCODED)` Query Request Contains: `fullname=Jane%20Doe` Result: Returns the entire query string encoded: `fullname=Jane%20Doe`
`$(QUERY_STRING_ DECODED{`*parameter*`})`	Not Applicable	Dictionary/ String	The same as `$(QUERY_ STRING{`*parameter*`})`	Variable Setting: `$(QUERY_STRING_ DECODED{CEO})` Query Request Contains: `fullname=Jane%20Doe` Result: Returns the value of `fullname` decoded. In this example, `fullname` returns a value of `Jane Doe`.

Exceptions and Errors

ESI uses several mechanisms to handle exceptions encountered during an ESI fragment request. In a given situation, you can make use of all mechanisms simultaneously, or use one at a time, depending on the business logic you are developing.

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

- The `alt` attribute of the `<esi:include>` tag

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

- The `try` | `attempt` | `except` block

- Default page in place of the fragment

When the fragment specified for the `src` attribute of the `<esi:include>` tag cannot be fetched, the fragment specified with the `alt` attribute is used as an alternative. If the `alt` attribute is not used or the fragment cannot be fetched, the `onerror` attribute is used. The `onerror` attribute is used before the `try` | `attempt` | `except` block. If the `try` | `attempt` | `except` block does not exist, the exception handling is propagated to the parent or template page. If all the ESI language controls fail, OracleAS Web Cache displays a default page in place of the fragment.

See Also:

- "ESI include Tag" on page 16-19
- "ESI try | attempt | except Tags" on page 16-28
- "Configure Error Pages" on page 8-37 for instructions on configuring a default fragment page

ESI Tag Descriptions

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

- ESI choose | when | otherwise Tags
- ESI comment Tag
- ESI environment Tag
- ESI include Tag
- ESI inline Tag
- ESI invalidate Tag
- ESI remove Tag
- ESI try | attempt | except Tags
- ESI vars Tag
- ESI <!--esi-->Tag

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.

- OracleAS 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, 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 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 and evaluates to null, then the expression evaluates to true; if name is not empty and does not evaluate to null, then the expression evaluates to false.

  > **Note:** If a variable exists but evaluates to an empty string, then the value is not considered null.

  ```
  <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 Boolean operators:

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

  The following expressions show incorrect usage of Boolean operators:

  ```
  (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. Example 16–2 shows invalid placement of statements.

Example 16–2   Statement Placement

```
<esi:choose>
  This markup is invalid because any characters other than whitespace
  are not allowed in this area.
  <esi:when test="$(HTTP_HOST) == 'www.company.com'">
    <esi:include src="/company.html" />
  </esi:when>
    This markup is invalid because any characters other than whitespace
    are not allowed in this area.
  <esi:when test="$(HTTP_COOKIE{fragment) == 'First Fragment'">
    <img src="/img/TheImage.gif">
  </esi:when>
    This markup is invalid because any characters other than whitespace
    are not allowed in this area.
   <esi:otherwise>
     The default selection.
  </esi:otherwise>
   This markup 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 comment Tag

The `<esi:comment>` tag enables you to comment ESI instructions, without making the comments available in the 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 OracleAS Web Cache. If comments need to be visible in the HTML output, use standard XML/HTML comment tags.

Example

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

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

ESI 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

There are two forms of this tag. In the first form, `<esi:environment>` does not have a closing `</esi:environment>` tag:

```
<esi:environment src="environment_URL" name="environment_name"
[max-age="expiration_time [+ removal_time]]" [method="GET|POST"]
[onerror="continue"] [timeout="fetch_time"]/>
```

In the second form with elements, `<esi:environment>` has a closing `</esi:environment>` tag:

```
<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" value="value"/>]
  [<esi:request_body value="value"/>]
  [<esi:log>log_message</esi:log>]
</esi:environment>
```

Attributes

- `src`—Specifies the URL from which to obtain environment variables and their values.

  The URL can be either an absolute or relative URL. When specifying an absolute URL, use one of the following formats:

  - `"http://host_name:port/path/filename"`

  - `"https://host_name:port/path/filename"`

  If you specify the host name for an absolute URL, you must prefix it with `http://` or `https://`. An HTML parser treats the `host:80` in the following URL as a folder name rather than a host name:

  ```
  src="host:80/index.htm"
  ```

  To make this URL valid, you specify the following:

  ```
  src="http://host:80/index.htm"
  ```

  Relative URLs are resolved relative to the template page. The result sets the ESI environment for the current template.

  The source code of the URL requires the following XML format:

  ```
  <?xml version="1.0"?>
  <esi:environment esiversion="ORAESI/9.0.4">
    <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 environment fragment. 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 OracleAS Web Cache to use.

- log—Specifies a log message of the fragment to be included in the *access_log_ file*.fragment file when the x-esi-info log field is set. You can provide a descriptive text string that identifies the fragment and the application that generated the fragment. By providing descriptive text, you can easily identify the fragment in the log file, enabling you to determine how often the fragment is requested.

  > **See Also:** Table 15–6 on page 15-16 for further information about the x-esi-info log field

Syntax Usage

- Specify only one <esi:environment> tag for each template page, before other ESI tags.

- The attributes do not need to be in a particular order.

- Do not specify more than one request_body element.

- You can have zero or more request_header elements.

  Use multiple request_header elements to specify multiple HTTP request header fields:

```
<esi:environment src="environment_URL"
   [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:environment>
```

- If no request_header elements are specified, OracleAS Web Cache uses other request headers from the parent page.

- Do not specify more than one log element.

  > **See Also:**
  >
  > - "Variable Expressions" on page 16-5 for usage instructions for variables
  >
  > - "ESI include Tag" on page 16-19 for usage notes on max-age, method, onerror, request_body, and request_header
  >
  > - "Exceptions and Errors" on page 16-9 for usage notes on onerror

Example

The following ESI output specifies `logindata` to refer to the environment variables stored in `catalog.xml`. The file `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})"/>
```

The file `catalog.xml` has the following content:

```
<?xml version="1.0"?>
<esi:environment esiversion="ORAESI/9.0.4">
  <product_description>stereo</product_description>
  <vendorID>3278</vendorID>
  <partner1>E-Electronics</partner1>
  <partner2>E-City</partner2>
</esi:environment>
```

The following ESI output specifies `logindata` to refer to the environment variables stored in `env.dat`. The file `env.dat` enables access to the value of the `env` environment variable, which is used as a parameter in the included log message for `dir1.txt`. The log messages for `dir1.txt` and `esi-log2.html` are written to the *access_log*.`fragment` file when the `x-esi-info` log field is set and the fragments are requested.

```
<esi:environment src="/esi/env.dat" name="env">
  <esi:log>Used environment /esi/env.dat</esi:log>
</esi:environment>

<esi:include src="/cached/dir1.txt">
  <esi:log>Fragment:/cache/dir1.txt is included, by $(env{xl_name})</esi:log>
</esi:include>

<font color="red">Including /cgi-bin/esi-fetch.sh?/esi/esi-log2.html in
esi-log1.html </font>
<esi:include src="/cgi-bin/esi-fetch.sh?/esi/esi-log2.html">
  <esi:log>Fragment: /cgi-bin/esi-fetch.sh?/esi/esi-log2.html is included
 </esi:log>
```

ESI include Tag

The `<esi:include>` tag provides syntax for including fragments.

> **See Also:** "Fragmentation with the Inline and Include Tags" on page 2-17 for a comparison of `<esi:inline>` and `<esi:include>` usage

Syntax

There are two forms of this tag. In the first form, `<esi:include>` does not have a closing `</esi:include>` tag:

```
<esi:include src="URL_fragment" [alt="URL_fragment"]
[max-age="expiration_time [+removal_time]]" [method="GET|POST"]
[onerror="continue"] [redirect=yes|no] [timeout="fetch_time"]/>
```

In the second form, with elements, `<esi:include>` has a closing `</esi:include>` tag:

```
<esi:include src="URL_fragment" [alt="URL_fragment"]
   [max-age="expiration_time[+removal_time]"] [method="GET|POST"]
   [onerror="continue"] [redirect=yes|no] [timeout="fetch_time"]>
 [<esi:request_header name="request_header" value="value"/>]
 [<esi:request_body value="value"/>]
 [<esi:log>log_message</esi:log>]
</esi:include>
```

Attributes

- `src`—Specifies the URL of the fragment to fetch. The URL can be a literal string or it can include variables.

  The URL can either be an absolute or relative URL. When specifying an absolute URL, use one of the following formats:

  - `"http://host_name:port/path/filename"`

  - `"https://host_name:port/path/filename"`

  If you specify the host name for an absolute URL, you must prefix it with `http://` or `https://`. An HTML parser treats the `host:80` in the following URL as a folder name rather than a host name:

  ```
  src="host:80/index.htm"
  ```

  To make this URL valid, you specify the following:

  ```
  src="http://host:80/index.htm"
  ```

  Relative URLs are resolved relative to the template page. The included result replaces the element in the markup served to the browser.

  You can specify an XML fragment as long as the XML file fragment is valid XML. For example, the following specifies that OracleAS Web Cache use XSL Transformations (XSLT) to transform the XML into HTML using a stylesheet. The stylesheet maps XML formats to HTML formats:

  ```
  <?xml version="1.0"?>
  <?xml-stylesheet type="text/xml" href="stylesheet.xsl"?>
  ```

Ensure that both the XML fragment and the XSL stylesheet response pages are configured with a `Content-Type` response-header field that includes text and XML media types. For example:

```
Content-Type: text/xml
```

> **See Also:** http://www.xslt.com/ for complete information about XSLT

- `alt`—Specifies an alternative resource if the `src` is not found. The requirements for the value are the same as those for `src`.

- `max-age`—Specifies the time, in seconds, to expire the fragment, and optionally, specifies the time, in seconds, to remove the fragment after 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 fragment. 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.

- `redirect`—Specifies how to serve the fragment when the `src` fragment resides temporarily under a different URL. `yes` specifies that the URL be redirected and displayed; `no` specifies that the fragment URL not be redirected and an HTTP `302 Found` status code be served in place of the fragment. `yes` is the default.

- `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.

> **See Also:** "Exceptions and Errors" on page 16-9 for usage notes on `alt` and `onerror`

Elements

- `request_body`—Specifies the HTTP request body of the fragment.

- `request_header`—Specifies an HTTP request header field and value for OracleAS 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.

- `log`—Specifies a log message of the fragment to be included in the *access_log*.fragment file when the `x-esi-info` log field is set. You can provide a descriptive text string that identifies the fragment and the application that generated the fragment. By providing descriptive text, you can easily identify the fragment in the log file, enabling you to determine how often the fragment is requested.

> **See Also:** Table 15–6 on page 15-16 for further information about the `x-esi-info` log field

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` attribute supports both HTTP and HTTPS. OracleAS Web Cache permits the template and fragments to use different protocols. Note the following:

  - If the `src` attribute specifies a fragment's relative path, such as `src="/PersonalizedGreeting"`, 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 (**Origin Servers, Sites, and Load Balancing** > **Site-to-Server Mapping**) of OracleAS Web Cache Manager, then OracleAS Web Cache uses the protocol configured for the origin server in the Site-to-Server Mapping page. OracleAS Web Cache also reports the following warning message to the event log:

    ```
    [Date] [warning 11250] [ecid: request_id, serial_number]
    ESI include fragment 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:80/gifs/frag1.gif"/>` and the site-to-server mapping specifies HTTP for the origin server, then `http://www.company.com:80/gifs/frag1.gif` is used and the following message appears in the event log:

    ```
    [03/Feb/2005:23:16:46 +0000] [warning 11250] [ecid: 90125204378,0]
    ESI include fragment protocol does not match origin server protocol:
    Origin Server Protocol=http URL=https://www.company.com:80/gifs/frag1.gif
    ```

- Do not specify more than one `request_body` element.

- You can have zero or more `request_header` 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>
  ```

- Do not specify more than one `log` element.

Usage

The `<esi:include>` tag instructs OracleAS Web Cache to fetch the fragment specified by the `src` attribute.

If the include is successful, the contents of the fetched `src` URL are displayed. 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.

The `max-age` control directive in the `Surrogate-Control` response-header field applies to the response; the `max-age` attribute applies only to that particular usage of the fragment response through the `<esi:include>` tag. If both the `max-age` control directive in the `Surrogate-Control` response-header field and the `max-age` attribute are set, then the effective expiration and removal time-to-live for this particular inclusion are the longest maximum age of the expiration and the removal time-to-live, respectively. If a particular page has a greater tolerance for staleness of a fragment, then set 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 object never expires.

If `method` is not set, then `GET` is assumed. However, if the `request_body` element is set, then `POST` is assumed.

OracleAS Web Cache generates the following HTTP request headers for all fragment requests:

- `Host`
- `Content-Length`
- `Surrogate-Capability`
- `Connection`

The `request_header` element enables you to control HTTP headers other than these. Do not specify these HTTP request headers as `request_header` attributes, as a conflict can affect the operation of OracleAS Web Cache.

If no `request_header` elements are specified, OracleAS Web Cache uses other request headers from the parent page.

> **See Also:** "Fragmentation with the Inline and Include Tags" on page 2-17 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, OracleAS Web Cache ignores the includes and serves the page. If the fetch succeeds, OracleAS Web Cache includes the fragment. OracleAS 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 an HTTP request body that includes the date. Log message `"Fragment: /Personalized Greeting is included"` is written to the *access_log*.fragment file when the `x-esi-info` log field is set and the fragment is requested.

```
<esi:include src="/PersonalGreeting">
  <esi:request_header name="Cookie" value="pname=Scott Tiger"/>
  <esi:request_body value="day=05, month=10, year=2001"/>
  <esi:log>Fragment: /Personalized Greeting is included</esi:log>
</esi:include>
```

> **See Also:** "Example of a Portal Site Implementation" on page 12-35 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. OracleAS Web Cache stores and assembles these fragments independently as `<esi:include>` fragments.

> **See Also:** "Fragmentation with the Inline and Include Tags" on page 2-17 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` instructs OracleAS Web Cache to fetch a fragment from the origin server when it expires. The template for the fragment is not included during this fetching process. `no` instructs OracleAS Web Cache to fetch the entire template from the origin server when there is a cache miss, and then try to extract all the fragments from the template.

- `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 OracleAS Web Cache the way `<esi:include>` fragments are. When a non-fetchable fragment is needed by OracleAS Web Cache, OracleAS 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, OracleAS 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, the response returned to the browser is undefined.

> **See Also:**
>
> - "Using Inline for Non-Fetchable Fragmentation" on page 2-17
>
> - "Using Inline for Fetchable Fragmentation" on page 2-18
>
> - "ESI include Tag" on page 16-19 for usage notes on the `maxage` attribute

Example

The following ESI output embeds financial headlines:

```
<esi:inline name="/Top_News_Finance">
Latest News for Finance
<TABLE>
  <TR>
    Blue-Chip Stocks Cut Losses; Nasdaq Up MO
    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 of a Portal Site Implementation" on page 12-35 for an extended example of `<esi:inline>` usage

ESI invalidate Tag

The `<esi:invalidate>` tag enables you to configure an invalidation request within the response of a browser page.

Syntax

Basic invalidation syntax:

```
<esi:invalidate [output="yes"]>
 <?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>
</esi:invalidate>
```

Advanced invalidation syntax:

```
<esi:invalidate [output="yes"]>
 <?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|SEARCHKEY"
                TYPE="SUBSTRING|REGEX"
                VALUE="value"/>
        </ADVANCEDSELECTOR>
     <ACTION REMOVALTTL="TTL"/>
     <INFO VALUE="value"/>
     </OBJECT>
 </INVALIDATION>
</esi:invalidate>
```

Attributes

- `output`—`yes` specifies that the invalidation result be included in the browser response, enclosed within comments `<!--result-->`. `no` specifies that the invalidation result not be displayed in the output. Specify a value of `yes` for a test environment; specify a value of `no` for a production environment.

Usage

See: "Inline Invalidation in HTTP Responses" on page 13-31

Example

See: "Example: Using Inline Invalidation" on page 13-32

ESI remove Tag

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

Syntax

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

Usage

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

Example

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

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

Normally, when OracleAS 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 the browser, which ignores ESI markup. However, the browser displays the `<A HREF=...>` HTML link.

ESI try | attempt | except Tags

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

> **See Also:** "Exceptions and Errors" on page 16-9 for usage notes on `alt` and `onerror`

Syntax

In the following form, only one `<esi:except>` tag is supported:

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

In the following form, multiple `<esi:except>` tags with different types are supported:

```
<esi:try>
  <esi:attempt>
    Try this...
  </esi:attempt>
  <esi:except [type="type"]>
    If the attempt fails, then perform this action...
  </esi:except>
  <esi:except [type="type"]>
    Perform this action...
  </esi:except>
  <esi:except>
    If the attempt fails, then perform this action...
  </esi:except>
</esi:try>
```

Usage

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

Specify an `<esi:except>` tag without a type for general errors; specify an `<esi:except>` tag with a type for specific errors. The `<esi:except>` tag accepts the following case-insensitive types:

- `nestingtoodeep`: An error occurs because the fragment include depth has exceeded the maximum include depth.

- `originserverbusy`: An error occurs because the origin server for this fragment is busy and cannot accept new requests now. This is caused by OracleAS Web Cache-to-origin server request queue limit being reached.

- `noconnection`: An error occurs because the cache is unable to connect to the origin server serving this fragment.

- `networktimeout`: An error occurs because a fragment request to the origin server has timed out in the network connection.

- `httpclienterror`: An error occurs because the origin server returns an HTTP 4*xx* status code, a client error, such as a malformed HTTP request or an unauthorized access.

- `httpservererror`: An error occurs because the origin server returns an HTTP 5*xx* status code, a server error.

- `incompatiblefragmentversion`: An error occurs because a fragment's processing requirement is not supported or not compatible with the template. `<!-- WEBCACHETAG-->` and `<!-- WEBCACHEEND-->` processing in an ESI fragment is not compatible with ESI processing. A fragment may be plain data that does not need any processing in the cache, or it may be an ESI template itself that requires processing of ESI features supported in this release. The ESI features in use are specified by the `Surrogate-Control content` control directive.

- `incorrectresponseheader`: An error occurs because the response headers for a fragment causes the error.

- `incorrectesifragment`: An error occurs when OracleAS Web Cache tries to parse or process the ESI fragment response body due to errors in the body.

- `incorrectxmlfragment`: An error occurs because there is an error in XSLT retrieval, parsing, or processing by OracleAS Web Cache.

Example

The following ESI markup attempts to fetch an advertisement. If the advertisement cannot be included, OracleAS 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>
```

The following ESI markup attempts to fetch a fragment. If the fragment cannot be included because of `httpclienterror`, then OracleAS Web Cache includes `/cgi-bin/esi-fetch?/esi/tryNestL1.html` instead.

```
<esi:try>
  <esi:attempt>
    <esi:include src="/frag.html"/>
  </esi:attempt>
  <esi:except type="httpclienterror">
    <esi:include src="/cgi-bin/esi-fetch?/esi/tryNestL1.html"/>
  </esi:except>
</esi:try>
```

The following `<esi:try>` attempts to include the fragment `http://server.portal.com/pls/ppcdemo/!PCDEMO.wwpro_app_provider.execute_portlet/513104940/26` containing several HTTP request headers. If the fragment cannot be included because of various `type` errors, OracleAS Web Cache returns an `Unknown ESI Exception` error.

```
<esi:try>
  <esi:attempt>
    <esi:include src="http://server.portal.com/pls/ppcdemo/!PCDEMO.wwpro_app_
provider.execute_portlet/513104940/26" timeout="15000" >
       <esi:request_header name="X-Oracle-Device.MaxDocSize" value="0"/>
       <esi:request_header name="Accept"
       value="text/html,text/xml,text/vnd.oracle.mobilexml"/>
       <esi:request_header name="User-Agent"
         value="Mozilla/4.0 (compatible; MSIE 5.5; Windows; YComp 5.0.0.0)
         RPT-HTTPClient/0.3-3"/>
       <esi:request_header name="Device.Orientation" value="landscape"/>
       <esi:request_header name="Device.Class" value="pcbrowser"/>
       <esi:request_header name="PORTAL-SUBSCRIBER" value="us"/>
       <esi:request_header name="Device.Secure" value="false"/>
       <esi:request_header name="PORTAL-SUBSCRIBER-DN"
          value="dc=us,dc=oracle,dc=com"/>
       <esi:request_header name="PORTAL-SUBSCRIBER-GUID"
          value="A5EE385440E6252BE0340800208A8B00"/>
       <esi:request_header name="Accept-Language" value="en-us"/>
       <esi:request_header name="PORTAL-USER-DN"
          value="cn=public,cn=users,dc=us,dc=oracle,dc=com"/>
       <esi:request_header name="PORTAL-USER-GUID"
          value="A5EE55B396E22651E0340800208A8B00"/>
       <esi:request_header name="Content-Type"
          value="application/x-www-form-urlencoded"/>
    </esi:include>
  </esi:attempt>
  <esi:except type="incompatiblefragmentversion" >
     This happens when a fragment's processing requirement is not supported
     or not compatible with the template.
  </esi:except>
  <esi:except type="noconnection" >
     The cache is unable to connect to the origin server serving this fragment.
  </esi:except>
  <esi:except type="nestingtoodeep" >
     The fragment include depth has exceeded the maximum include depth. The
     default value defined in Web Cache is 3.
  </esi:except>
  <esi:except type="httpservererror" >
     The origin server returns an HTTP 5xx status code, a server error.
  </esi:except>
  <esi:except type="httpclienterror" >
     The origin server returns an HTTP 4xx status code, a client error, such as
     a malformed HTTP request or an unauthorized access.
  </esi:except>
  <esi:except type="incorrectresponseheader" >
     This happens when the response headers for a fragment cause the error.
  </esi:except>
  <esi:except type="incorrectxmlfragment" >
     This happens when there is any kind of error in OracleAS Web Cache XSLT
     retrieval, parsing, or processing.
  </esi:except>
  <esi:except type="originserverbusy" >
     The origin server for this fragment is busy and cannot accept new requests
     now. This is caused by OracleAS Web Cache-to-origin server request queue
     limit.
  </esi:except>
  <esi:except type="networktimeout" >
      This is thrown by a fragment whose request to the origin server has timed
      out in the network connection.
```

```
        </esi:except>
        <esi:except type="incorrectesifragment" >
            An error is encountered when OracleAS Web Cache tries to parse or process
            the ESI fragment response body due to errors in the body.
        </esi:except>
        <esi:except>
            Unknown ESI Exception
        </esi:except>
</esi:try>
```

ESI vars Tag

The `<esi:vars>` tag enables you to use variables outside of ESI tags. For example, instead of specifying a variable inside a `<esi:include>` or `<esi:choose>` block, you can use the `<esi:vars>` tag to specify a variable inside HTML code.

Syntax

```
<esi:vars>Optional HTML code $(VARIABLE_NAME{key}) Optional HTML code</esi:vars>
```

Syntax Usage

- If the variable does not use the complete `$(VARIABLE_NAME{key})` format, OracleAS Web Cache reports the following error message to the event log:

  ```
  [Date] [error 12086] [ecid: request_id, serial_number]
  ESI syntax error. Unrecognized keyword keyword is at line line.
  ```

- Do not nest the `<esi:vars>` tag within an HTML code line. The following is an example of incorrect syntax:

  ```
  HTML code <esi:vars>$(VARIABLE_NAME{key})</esi:vars>HTML code
  ```

  For example, the following is invalid:

  ```
  <IMG SRC="http://www.example.com/<esi:vars>$(HTTP_
  COOKIE{type})</esi:vars>/hello.gif"/>
  ```

Usage

> **See Also:** "Variable Expressions" on page 16-5 and "ESI environment Tag" on page 16-16 for usage of HTTP request variables 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>
```

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, OracleAS 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

```
In the following ESI markup, the <!--esi and --> are removed in the final
output. The output displays the content generated by
<p><esi:vars>Hello, $(HTTP_COOKIE{name})!</esi:vars></p>,
plus any surrounding text.

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

If the ESI markup cannot be processed, then the <p><esi:vars>Hello, $(HTTP_COOKIE{name})!</esi:vars></p> is displayed in the HTML output.

17

Event Log Messages

This chapter contains a complete listing of the event logs messages, except those with a severity level of debug or internal error. It contains these topics:

- Message Format
- Messages that Reference Network Security Messages (NZE)
- Message Listing

Message Format

OracleAS Web Cache rates each message by severity level:

- `notification`: A normal activity, for example, the start up or shut down of the cache server or an event log rollover.

- `trace`: Information that helps trace the activity of OracleAS Web Cache. These messages are particularly helpful in debugging caching rules. Messages with a severity level of trace are written to the event log file only when a Verbosity of Trace is selected in the event log configuration.

- `warning`: An event which is abnormal or potentially a sign of a problem, but OracleAS Web Cache will continue to operate without any lack of service. For example, a request header included a malformed header, but OracleAS Web Cache interpreted it in some way and continued processing.

- `error`: An event which can cause lack of service only within the request/response transaction. For example, OracleAS Web Cache cannot continue the transaction because of malformed request.

- `alert`: An event which can affect an OracleAS Web Cache instance and can cause lack of service in a subsequent request/response transaction. For example, the origin server is not reachable or sufficient memory is unavailable. In many cases, the cache server will be shut down when an alert event is encountered.

In addition, OracleAS Web Cache provides messages, that are not included in this appendix, with severity levels of debug or internal error:

- `internal error`: Contact Oracle Support for assistance.
- `debug`: Intended for use only by Oracle Support.

Messages that Reference Network Security Messages (NZE)

Several SSL-related errors are reported with NZE errors in the event log. *Oracle Database Error Messages* lists these errors in full, and "Resolving Common NZE Errors" on page E-19 describes the most commons NZE errors reported to the event log.

Message Listing

00800 Auto-Restart started successfully

notification

Cause: The Auto-Restart process has successfully completed its initialization.

Action: No action required.

08000 Auto-Restart process terminating.

alert

Cause: This error usually occurred as a result of insufficient system resources.

Action: If the problem persists, contact Oracle Support.

08001 Failed to start the cache server. Auto-Restart process terminating.

alert

Cause: This error usually occurred as a result of insufficient system resources.

Action: If the problem persists, contact Oracle Support.

08002 Failed to find a Web Cache listening port that does not require SSL client-side certificates. Auto-Restart pinging is disabled.

warning

Cause: Auto-Restart does not support the use of client-side certificates. If all OracleAS Web Cache listening ports require client-side certificates, then pinging is disabled in the Auto-Restart mechanism.

Action: To enable pinging, add a listening port that does not require client-side certificates.

08500 Cache server not responding on port %d.

error

Cause: The cache server may not have been running. The Auto-Restart process will continue to try to reach the cache server on the specified port.

Action: No action required.

08501 Timeout occurred communicating with the cache server.

error

Cause: The cache server was either in an unstable state or slow responding to queries.

Action: From the Auto-Restart page of Oracle Enterprise Manager or OracleAS Cache Manager, check the ping timeout interval and increase it if necessary.

08502 Lost connection with the cache server.

error

Cause: There were communication errors between the Auto-Restart process and the cache server. The Auto-Restart process will continue to try to reach the cache server.

Action: No action required.

08503 Operating system level error in %s() (%d): %s.

error

Cause: The specified system call returned the specified error.

Action: No action required.

08504 %u consecutive errors pinging the cache server.

notification

Cause: The Auto-Restart process received the specified number of failures when attempting to reach the cache server.

Action: No action required.

08505 %u consecutive error(s) pinging the cache server. %d error(s) required for restart.

notification

Cause: The Auto-Restart process received the specified number of failures when attempting to reach the cache server. The error message indicates the number of times the Auto-Restart process will attempt to reach the cache server.

Action: No action required.

08506 Unable to retrieve the cache server process ID from %s. Stopping the Auto-Restart process.

alert

Cause: Unable to read the OracleAS Web Cache server pid file. Most likely the file was deleted during a normal termination of the server. As a result, the Auto-Restart process was also shut down.

Action: No action required.

08507 Cache server process not running.

notification

Cause: There was no process on the system matching the process ID listed in the OracleAS Web Cache server pid file.

Action: No action required.

08508 Restarting the cache server.

alert

Cause: See accompanying error messages for more information.

Action: No action required.

08509 Starting the cache server.

alert

Cause: See accompanying error messages for more information.

Action: No action required.

08510 SSL set hardware acceleration failed. NZE-%d.

error

Cause: The SSL library failed to initialize the hardware SSL acceleration. The specified error message provides more detail.

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

08511 SSL Set Thread Usage Failed. NZE-%d.

error

Cause: The SSL library failed to set the threading model. The specified error message provides more detail.

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

08512 SSL Engine Failed to Initialize. NZE-%d.

error

Cause: The SSL engine failed to initialize. The specified error message provides more detail.

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

09000 Process %d exit(%d) at %d:%s [%s]

alert

Cause: The Web Cache process shutdown.

Action: No action required.

09001 An error occurred while disabling core file creation.

warning

Cause: The system/library call to disable core file creation returned an error.

Action: If the problem persists, contact Oracle Support.

09200 An error occurred while trying to create a background process.

alert

Cause: Could not open standard input while OracleAS Web Cache is creating a process.

Action: Determine why OracleAS Web Cache cannot read standard input or write to standard output for null device.

09403 Maximum number of file/socket descriptors set to %s.

notification

Cause: The specified number of file descriptors is set for use by OracleAS Web Cache.

Action: No action required.

09521 OracleAS Web Cache fails to find user (%s) in /etc/passwd.

alert

Cause: The specified user was not found in the password file.

Action: Add user to the password file.

09522 OracleAS Web Cache fails to find group (%s) in /etc/group.

alert

Cause: The specified group was not found in the group file.

Action: Add the group to the group file or modify the group ID in the Security page of Oracle Enterprise Manager or the Process Identity page of OracleAS Web Cache Manager.

09523 Invalid user ID (%s).

alert

Cause: The specified user ID was invalid.

Action: Correct the user ID in the Security page of Oracle Enterprise Manager or Process Identity page of OracleAS Web Cache Manager.

09524 Invalid group ID (%s)

alert

Cause: The specified group ID was invalid.

Action: Correct the group ID in the Security page of Oracle Enterprise Manager or Process Identity page of OracleAS Web Cache Manager.

09525 Permission denied when setting user ID (%s).

alert

Cause: The specified user did not have the required permissions when setting the user ID.

Action: Grant the user required permissions or log in as a different user.

09526 Permission denied when setting group ID (%s).

alert

Cause: The user did not have the required permissions when setting group ID.

Action: Grant the user required permissions or log in as a different user.

09534 Unexpected File I/O Error (%s) when opening file %s.

alert

Cause: Could not open specified file.

Action: Review specified error to determine the problem on your system.

09570 Directory not found when unlinking file %s

notification

Cause: The directory was not found.

Action: Verify that OracleAS Web Cache was installed properly. You may need to reinstall OracleAS Web Cache.

09571 Too many files open when unlinking file %s.

notification

Cause: Too many files were open for available system resources.

Action: Adjust system resources and restart OracleAS Web Cache.

09572 No disk space available when unlinking file %s

notification

Cause: Insufficient disk space was available on the system.

Action: The system is running low on disk space. Free up disk space and restart OracleAS Web Cache.

09573 Unexpected File I/O Error (%s) when unlinking file %s.

notification

Cause: There was an unexpected input/output file error.

Action: Review the specified error and correct the problem on the system.

09574 Permission denied when unlinking file %s.

notification

Cause: Permission denied on file.

Action: Set correct permission on the specified file.

09600 SIGTERM caught - program will shut down once all connections are complete.

notification

Cause: Normal notification behavior. Shutdown event written to the log file.

Action: No action required.

09601 SIGUSR1 caught - program will reread configuration files after all connections are complete.

notification

Cause: Normal notification behavior. Configuration file reread event written to the log file.

Action: No action required.

09602 All configuration files have been reread.

notification

Cause: Normal notification behavior. All configuration files have been reread and this configuration reload event written to the log file.

Action: No action required.

09603 There was an error rereading the configuration files. The program is exiting.

alert

Cause: Invalid OracleAS Web Cache configuration files.

Action: Correct the OracleAS Web Cache configuration files and restart OracleAS Web Cache.

09604 Could not increase the number of file/socket descriptors to %s.

alert

Cause: Ran out of file descriptors on the system.

Action: Decrease the Maximum Incoming Connections value in the OracleAS Web Cache Manager, or increase the system file descriptors. Run OracleAS Web Cache as the root user.

09605 Cannot create OPMN shutdown thread. Continuing with startup.

notification

Cause: OracleAS Web Cache could not create OPMN shutdown thread.

Action: Use OPMN to abort the OracleAS Web Cache process during OracleAS Web Cache shutdown.

09606 Cannot create OPMN shutdown event. Continuing with startup.

notification

Cause: OracleAS Web Cache could not create OPMN shutdown event.

Action: Use OPMN to abort the OracleAS Web Cache process during OracleAS Web Cache shutdown.

09607 The admin server started successfully.

notification

Cause: The OracleAS Web Cache admin server process started successfully.

Action: No action required.

09608 The cache server process started successfully.

notification

Cause: The OracleAS Web Cache server process has started successfully.

Action: No action required.

09609 The server process could not initialize.

alert

Cause: The OracleAS Web Cache configuration is invalid.

Action: Review the error log file or use the Event Viewer to help determine the cause of the problem.

09610 The server is exiting.

notification

Cause: The cache server or admin server process is terminating.

Action: Review the error log file or use the Event Viewer to help determine the cause of the problem.

09611 The server could not start the service thread.

alert

Cause: The server process could not start the Service thread.

Action: Determine why OracleAS Web Cache cannot start a thread on your system.

09612 OracleAS Web Cache %s, %s

notification

Cause: Normal notification behavior. The OracleAS Web Cache version number was written to the log file.

Action: No action required.

09614 The following OracleAS Web Cache internal files are pre-populated to the cache: %s

notification

Cause: The specified OracleAS Web Cache internal files are loaded into the cache.

Action: No required action.

09700 Unable to initialize NLS. Error %d

alert

Cause: A system/library call has unexpectedly failed.

Action: If the problem persists, contact Oracle Support.

09701 Unable to initialize NLS global structure.

alert

Cause: A system/library call has unexpectedly failed.

Action: If the problem persists, contact Oracle Support.

09702 NLS boot file not found or missing; using default.

warning

Cause: A system/library call has unexpectedly failed.

Action: If the problem persists, contact Oracle Support.

09703 Stop Issued. The program will shut down after all accepted requests are served, or a timeout occurs.

notification

Cause: Stop was issued.

Action: OracleAS Web Cache will be stopped.

09704 Abort Issued. The program will shut down immediately.

notification

Cause: Abort was issued.

Action: OracleAS Web Cache will be stopped immediately.

09705 Error accessing the Oracle Notification Service library, %s: %s

alert

Cause: An error occurred when attempting to open or access functions in the Oracle Notification Service (ONS) library.

Action: See the operating system level error message included in this message. The error message provides further details, such as "file not found" or "function not available."

09706 Ignoring unrecognized option: %s

alert

Cause: An alert was logged.

Action: Refer to the logged alert details and take the appropriate action.

09707 Failed to start the server.

alert

Cause: Failed to start the OracleAS Web Cache server. This error usually occurs as a result of insufficient system resources or invalid configuration.

Action: 1. Reduce the capacity of cluster members in the Cluster Members and Properties page of Oracle Enterprise Manager or the Clustering page of OracleAS Web Cache Manager. 2. Ensure the IP address and listening ports of OracleAS Web Cache conflict with existing applications. 3. Check whether OracleAS Web Cache has root privilege to bind to specified ports. You can use the webcache_setuser.sh script to run webcached as root. See section "Running webcached with Root Privilege" in the OracleAS Web Cache Administrator's Guide.

09709 OracleAS Web Cache fails HTTP initialization.

alert

Cause: OracleAS Web Cache failed to initialize HTTP.

Action: 1. Check the configuration of the access log in the Logging page of Oracle Enterprise Manager or the Access Logs page of OracleAS Web Cache Manager. 2. Check whether OracleAS Web Cache has root privilege to set the process ID. You can use the webcache_setuser.sh script to run webcached as root. See section "Running webcached with Root Privilege" in the OracleAS Web Cache Administrator's Guide. 3. If the problem persists, contact Oracle Support.

09710 Cache server fails to initialize.

alert

Cause: Insufficient system resources or invalid OracleAS Web Cache configuration.

Action: Review the error log file or use the Event Viewer to help determine the cause of the problem.

09711 Problem loading message file for event log. Exiting.

alert

Cause: Message file for event log was not correctly loaded.

Action: Check the existence and permissions of the file wxeus.msb in $ORACLE_
HOME/webcache/mesg/wxeus.msb on UNIX and ORACLE_
HOME\webcache\mesg on Windows.

09712 Unable to open the Oracle Notification Service library, %s:%s. Searching the library path.

warning

Cause: An error occurred when attempting to open the Oracle Notification
Service (ONS) library.

Action: See the operating system level error message included in this message.
The error message provides further details, such as "file not found." OracleAS Web
Cache will search the library path. If the library is still not found, then OracleAS
Web Cache returns event message 09705.

10054 Unable to resume service, UIPI return value is %s.

alert

Cause: Service could not be resumed.

Action: Review the error to determine the problem on your system.

10417 MIME Type Extension (%s) exceeds current size limit.

warning

Cause: Unexpected MIME type configured.

Action: No action required.

10418 Encoding Extension (%s) exceeds current size limit.

warning

Cause: Unexpected encoding type configured.

Action: No action required.

10419 Language Extension (%s) exceeds current size limit.

warning

Cause: Unexpected language type configured.

Action: No action required.

10536 You must specify a full path for %s. See %s %s.

alert

Cause: Failed to specify full path name.

Action: Specify a full path name for the file.

11058 Stale object updated for %s%s

trace

Cause: The complete response was successfully received and the cache entry is
updated with the new response.

Action: No action required.

11059 Garbage object updated for %s%s

trace

Cause: The complete response was successfully received and the cache entry is
updated with the new response.

Action: No action required.

11060 Object update aborted for %s%s.

trace

Cause: The response will not be used to update the stale or garbage object in the cache because it was determined that it is not cacheable.

Action: No action required.

11062 <!-- WEBCACHETAG--> and <!-- WEBCACHEEND--> tag parsing error

warning

Cause: The syntax for the <!-- WEBCACHETAG--> and <!-- WEBCACHEEND--> tags was incorrect.

Action: Correct the syntax of the <!-- WEBCACHETAG--> and <!-- WEBCACHEEND--> tags in the source.

11064 ESI object %s%s parsing error

warning

Cause: Object contained invalid ESI syntax.

Action: Correct the ESI syntax in the source.

11077 Cluster member sent invalid configuration error

warning

Cause: The cache cluster member sent a response to OracleAS 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.

Action: Compare the two configuration files, and then propagate the proper version to all cluster members.

11078 Malformed environment object %s%s

warning

Cause: The syntax of the XML file specified by the ESI environment is incorrect.

Action: Check the XML object for possible syntax errors.

11083 Proxy server configuration error: %s

error

Cause: The proxy server was not correctly configured.

Action: Check the proxy server configuration in the Origin Servers page of Oracle Enterprise Manager or OracleAS Web Cache Manager and the general configuration of the proxy server itself.

11085 Authentication for the remote or subscriber cache is being challenged by a central or provider cache. Retry.

trace

Cause: Access to the central or provider cache requires authorization.

Action: Try accessing the cache again.

11088 Following URL is now in cache:

trace

Cause: The complete response was successfully received and populated into the cache.

Action: No action required.

11089 Error while sending the entity body to the origin server

error

Cause: OracleAS Web Cache encountered a bad network connection or the origin server was down.

Action: No action required

11091 Error while processing ESI for %s%s

warning

Cause: An error was encountered while ESI processing was done for the response. The page will not be cached.

Action: No action required.

11092 Object validated for %s%s

trace

Cause: The cached object was validated by the origin server and its expiration timestamps were updated accordingly.

Action: No action required.

11093 Content-Length value %d sent by origin server does not match number of bytes received, that is %d bytes for %s%s.

warning

Cause: This error can occur if the origin server sent an incorrect Content-Length value or if the connection was unexpectedly terminated.

Action: Check that the origin server is sending the correct Content-Length for the URL mentioned.

11095 Update fails because object is no longer cacheable.

trace

Cause: This request was no longer cacheable.

Action: No action required.

11110 The maximum cache size is set too low. OracleAS Web Cache failed to initialize.

alert

Cause: The maximum cache size is set too low. Some of the data structures could not be initialized because of insufficient memory. This error could also be caused by changes to the file internal.xml.

Action: Increase the maximum cache size. If this fails, revert to the original values in internal.xml. Contact Oracle Support for information about the internal.xml file.

11111 Based on maximum cache memory limit %s, a cache table is created.

trace

Cause: Normal trace behavior.

Action: No action required.

11118 Duplicate search key %s found in Surrogate-Key response header is discarded.

error

Cause: A duplicate search key was discovered in the Surrogate-Key response header.

Action: Remove the duplicate search key token from Surrogate-Key response header.

11119 The number of search keys changed.

warning

Cause: The number of search keys has changed for the same cached object.

Action: No action required.

11128 A duplicate cache insertion is found. Cache insertion fails.

trace

Cause: Normal trace behavior.

Action: No action required.

11136 An error occurs during shutdown. The internal.xml file has been corrupted or moved.

alert

Cause: The internal.xml file was corrupted or moved during the shutdown operation.

Action: Recover internal.xml by copying from internal.xml.bak.

11201 OracleAS Web Cache is unable to resolve the IP address of %s. Check your DNS setup.

alert

Cause: OracleAS Web Cache was unable to resolve the IP address of the origin server.

Action: Check whether the origin server is a valid host. You can alter origin server configuration in the Origin Servers page of Oracle Enterprise Manager or OracleAS Web Cache Manager.

11202 Origin server %s does not use IPv4.

alert

Cause: OracleAS Web Cache only supports IP version 4. The IP address of the origin server could not be resolved because it used another version of IP.

Action: Check that the origin server is a valid host and check the DNS setup. You can alter origin server configuration in the Origin Servers page of Oracle Enterprise Manager or OracleAS Web Cache Manager.

11203 OracleAS Web Cache fails to connect to the origin server (id=%d).

warning

Cause: An origin server is not available.

Action: Check the status of the origin server and the network connection.

11205 OracleAS Web Cache marked the origin server %s:%u (id=%d) as down and removed all session information bound to it.

trace

Cause: The origin server was disabled or reached its failover threshold. OracleAS Web Cache automatically distributes the request load over the remaining origin servers and polls the failed origin server for its current status.

Action: No action required.

11206 OracleAS Web Cache is unable to start pinging the origin server. rc code=%d

alert

Cause: There was a potential memory allocation error.

Action: Check the configured memory usage in the Resource Limits and Timeouts page of Oracle Enterprise Manager or the Resource Limits page of OracleAS Web Cache Manager.

11207 Origin server %s:%u (id=%d) is down; OracleAS Web Cache starts polling for status.

warning

Cause: The origin server reached its failover threshold. OracleAS Web Cache automatically distributes the request load over the remaining origin servers and polls the failed origin server for its current status.

Action: No action required.

11208 Cache cluster member %s (id=%d) is down; cache cluster members start polling for status. Ping threshold set at %d.

warning

Cause: The cache cluster member reached its failover threshold. OracleAS Web Cache reassigns ownership of cache content, and polls the failed cache cluster member for its status.

Action: No action required.

11209 Origin server %s:%u (id=%d) is back up; OracleAS Web Cache stops polling and restores origin server.

warning

Cause: The ping URL to the origin server succeeded.

Action: No action required.

11210 Cache cluster member %s (id=%d) is up; cache cluster members stop polling and restore it.

warning

Cause: The ping URL to the cache cluster member succeeded.

Action: No action required.

11211 All origin servers for site %s are down. Last origin server was (id=%d).

warning

Cause: All origin servers configured for the site are down.

Action: No action required.

11212 Cache cluster member %s (id=%d) pinged successfully. %d successful pings so far.

warning

Cause: An origin server is not available.

Action: Check the status of the origin server and the network connection.

11220 Session binding routing to origin server %s:%u (id=%d) for site %s

trace

Cause: Normal trace behavior.

Action: No action required.

11221 The origin server request is queued for site %s and site-to-server mapping %s.

trace

Cause: The origin server was busy.

Action: No action required.

11222 The origin server request is dropped due to a full queue for site %s and site pattern %s.
trace

Cause: The origin server request queue was full.

Action: No action required.

11223 The origin server request is routed to peer cache cluster member %s (id=%d) for site %s.
trace

Cause: The request did not support session binding, and the requested object was owned by the peer cache cluster member.

Action: No action required.

11224 Site %s matches site-to-server mapping %s.
trace

Cause: Normal trace behavior.

Action: No action required.

11225 OracleAS Web Cache cannot find any running origin servers to route the request for site %s.
warning

Cause: All origin servers are down.

Action: No action required.

11226 The session binding request is to a busy origin server %s:%u (id=%d) for site %s.
trace

Cause: The session binding request was to a server was that was busy.

Action: No action required.

11227 Request is routed to origin server %s:%u using load balancing.
trace

Cause: Normal trace behavior.

Action: No action required.

11228 No defined origin server can serve this request for site %s. The request is dropped.
trace

Cause: This request was not an ESI template and could not be served by dynamic origin servers for ESI provider sites.

Action: No action required.

11229 This ESI fragment request is forwarded to a dynamic origin server for ESI provider site %s.
trace

Cause: The request was for an ESI fragment without a matching site-to-server mapping.

Action: No action required.

11240 Origin server %s:%u (id=%d) has %d connections timed out or closed for other reasons.

trace

Cause: Normal trace behavior.

Action: No action required.

11250 ESI include fragment protocol does not match origin server protocol: Origin Server Protocol=%s URL=%s%s%s

warning

Cause: The protocol used in the src attribute does not match the protocol specified in the Sites page of Oracle Enterprise Manager or Site-to-Server Mapping page of OracleAS Web Cache Manager. OracleAS Web Cache used the protocol configured for the origin server in the Sites or Site-to-Server Mapping page.

Action: Correct the protocol in the template for the fragment.

11260 Session binding cookie %s with cookie value %s exceeds limit %d.

warning

Cause: The value of the session binding cookie exceeded the allowed length limit for the Internal-Tracking session binding mechanism. limit in OracleAS Web Cache.

Action: Consider using another session binding mechanism, such as Cookie-Based, or consider modifying the application so that the length of the session binding cookie value does not exceed the limit.

11261 Session binding query string parameter name %s with value %s exceeds limit of %d bytes.

warning

Cause: The value of the session binding query string parameter exceeded the allowed length limit for the Internal-Tracking session binding mechanism.

Action: Consider using another session binding mechanism, such as Cookie-Based, or consider modifying the application so that the length of the session binding cookie value does not exceed the limit.

11290 DNS cache miss. Scheduling DNS task for host name %s

trace

Cause: The entry for the host name was stale or missing in OracleAS Web Cache's DNS cache. OracleAS Web Cache scheduled a DNS task to get the host name.

Action: No action required.

11291 DNS lookup task scheduling fails for host %s.

alert

Cause: Asynchronous task scheduling failed.

Action: Check the log of corresponding asynchronous task scheduling.

11292 DNS lookup succeeds for host %s.

trace

Cause: The DNS cache miss and lookup succeeded.

Action: No action required.

11293 OracleAS Web Cache fails to connect to dynamic origin server for ESI provider site and again performs a DNS lookup for host %s.

trace

Cause: There was a possible stale DNS cache entry.

Action: No action required.

11294 OracleAS Web Cache fails to connect to a dynamic origin server for ESI provider site with fresh DNS lookup result for host %s.

alert

Cause: There was a potential system DNS error or invalid host.

Action: 1. Check whether the host is valid. 2. In a deployment environment, try to connect to this host and port. If the connection succeeds, then check the system/network DNS setup.

11295 OracleAS Web Cache failed in DNS lookup for host %s.

warning

Cause: There was a potential system DNS error or invalid host.

Action: 1. Check whether the host is valid. 2. In a deployment environment, try to connect to this host and port. If the connection succeeds, then check the system/network DNS setup.

11296 Dynamic origin server name %s is too long.

error

Cause: Dynamic origin server name specified for an ESI fragment is longer than 255 characters.

Action: Check the host name that appears in src attribute of <esi:include> tag.

11297 Cached DNS look up result for host %s is used.

trace

Cause: Cached DNS look up result was used to establish a connection to a dynamic origin server.

Action: No action required.

11300 The attempt to create an entry in the cache failed due to low memory.

alert

Cause: Memory low during insertion.

Action: No action required.

11303 Request method is not GET nor POST nor PUT; tunneling.

trace

Cause: The HTTP method in the request is not one of GET, POST or PUT. OracleAS Web Cache will tunnel this request between the client and the origin server.

Action: No action required.

11304 Cache miss request.

trace

Cause: The cache did not have a object matching the request.

Action: No action required.

11306 Request is for a non-cacheable object, forwarding request to the backend.

trace

Cause: Normal trace behavior.

Action: No action required.

11308 Request may not be cacheable according to caching rule.

trace

Cause: Normal trace behavior.

Action: No action required.

11311 OracleAS Web Cache generates a HTTP 200 OK status code.

trace

Cause: The request completed successfully.

Action: No action required.

11312 invalid Oracle-ECID request header

error

Cause: OracleAS Web Cache was unable to parse the Oracle-ECID request header.

Action: Correct the syntax of the Oracle-ECID request header.

11313 The cache server reached the maximum number of allowed incoming connections. Listening is temporarily suspended on port %d.

alert

Cause: The cache server reached the maximum number of incoming connections specified in the Resource Limits and Timeouts page in Oracle Enterprise Manager or Resource Limits page or OracleAS Web Cache Manager.

Action: Try again later. If this message persists, increase the maximum incoming connections in the Resource Limits and Timeouts page of Oracle Enterprise Manager or OracleAS Web Cache Manager.

11314 Listening has resumed on port %d.

alert

Cause: The number of incoming connection count is under the threshold, enabling the cache server to accept new requests from browsers.

Action: No action required.

11315 OracleAS Web Cache generates a HTTP 202 Accepted status code.

trace

Cause: The request was accepted.

Action: No action required.

11316 The request is for an end-user performance monitoring GIF file.

trace

Cause: Normal activity for end-user performance monitoring.

Action: No action required.

11317 OracleAS Web Cache generates an HTTP 304 Not-Modified status code.

trace

Cause: OracleAS Web Cache sent an HTTP 304 Not-Modified status code to the browser.

Action: No action required.

11321 Connection from browser cannot be established.

error

Cause: The handshake with the browser failed.

Action: No action required.

11327 The request contains incorrect content.

error

Cause: The object request could not be processed because the request contained incorrect or malformed information. Typical reasons are invalid, incomplete or malformed header information. This error can also occur when referencing ESI-only objects from non-ESI requests.

Action: Check the event log to determine which information is incorrect.

11330 The request prefix %s%s matches an alias for configured site %s.

trace

Cause: The alias for the site matched the Host request header.

Action: No action required.

11331 Request matches configured site %s.

trace

Cause: The Host request header matched the site name.

Action: No action required.

11332 Failed to rewrite URI from '%s' to '%s' according to rewrite rule '%s'. The original URI will be used.

warning

Cause: URI rewrite failed. OracleAS Web Cache will use the original URI in subsequent operations without rewriting.

Action: Modify URL rewrite rules.

11333 URI is rewritten from %s to %s according to rewrite rule %s.

trace

Cause: URI rewrite was successful.

Action: No action required.

11334 User-Agent request header %s does not match any rule. OracleAS Web Cache remapped User-Agent to the default value %s.

trace

Cause: User-Agent request header did not match any rule.

Action: Manually modify user-agent rules in the webcache.xml file.

11335 User-Agent request header remapped from %s to %s according to mapping rule %s.

trace

Cause: User-Agent request header matched a rule.

Action: No action required.

11336 The request fails authorization check.

error

Cause: The request was not processed due to insufficient privileges or authorization rejection.

Action: Make sure you are authorized to perform the invalidation operation.

11337 Administration service unavailable. Try again later.

warning

Cause: There were too many administration requests.

Action: Ensure that only one user is the administrator user.

11338 URL is in the cache

trace

Cause: There was a matching entry in the cache for this request.

Action: No action required.

11339 OracleAS Web Cache determined the request might be cacheable based on existing cache policy.

trace

Cause: There was no matching cache entry for this request. However, based on existing cache policy, OracleAS Web Cache determined that it is likely to be cacheable.

Action: No action required.

11340 A new entry in the cache is created.

trace

Cause: OracleAS Web Cache created an entry in the cache for this request.

Action: No action required.

11342 OracleAS Web Cache determined the request might not be cacheable based on existing cache policy.

trace

Cause: There was no matching cache entry for this request. However, based on existing cache policy, OracleAS Web Cache determined that it is likely to be not cacheable.

Action: No action required.

11343 The requested site does not have a matching origin server mapping. A bad request error will be returned.

error

Cause: There was no matching site-to-server mapping for this non-ESI-fragment request.

Action: No action required.

11344 Returning a freshly cached object.

trace

Cause: The cached object will be returned because it is still fresh.

Action: No action required.

11345 The cached object is stale and this request will be served stale.

trace

Cause: The cached object is stale and OracleAS Web Cache decided to serve the stale copy to the request.

Action: No action required.

11346 The cached object is stale and an origin server will be contacted for a fresh copy.

trace

Cause: Because the cached object is stale, OracleAS Web Cache fetched a new version of the object to update the cache and satisfy this request with the new response.

Action: No action required.

11347 The cached object is garbage.

trace

Cause: The cached object is in garbage state, meaning that it could not be used to serve to the client.

Action: No action required.

11348 There is already another pending request for the same object being forwarded to the origin server for a fresh copy. This request will wait for the fresh copy.

trace

Cause: The cached object is in garbage state, and there is another request for the same object being forwarded to the origin server for a fresh copy to update the cache. Therefore, this request will wait until the fresh copy arrives.

Action: No action required.

11349 The request will be forwarded to the origin server to obtain a fresh copy of the object.

trace

Cause: The cached object was in garbage state, and there was no other request pending to update the cache. OracleAS Web Cache will forward this request to the origin server for a fresh copy of the object to update the cache and satisfy this request with the new response.

Action: No action required.

11350 The request contains no site information in the URL or Host header. The configuration of the default site %s will be used.

trace

Cause: The request did not contain a Host header, nor did it have an absolute URL that contained a site name. OracleAS Web Cache will use the default site in the configuration for this request.

Action: No action required.

11351 The complete site name is: %s

trace

Cause: The complete site name was resolved based on the configuration.

Action: No action required.

11352 There is no matching defined site or aliases in the configuration for this request.

trace

Cause: The configuration did not contain a matching site definition for the request. The site name will be taken from the request as is.

Action: No action required.

11354 Out of memory while processing the request. A service unavailable response is returned to the client. Client IP: %s

alert

Cause: OracleAS Web Cache failed to obtain enough memory while processing the request.

Action: Adjust the maximum cache size in the Resource Limits and Timeouts page of Oracle Enterprise Manager or Resource Limits page of OracleAS Web Cache Manager.

11355 Single request header length exceeds configured maximum. A forbidden error response is returned to the client. Client IP: %s

error

Cause: One of the headers in the request exceeded the configured maximum.

Action: Adjust the maximum individual header size limit in the Security page of Oracle Enterprise Manager or OracleAS Web Cache Manager.

11356 Total request header length exceeds configured maximum. A forbidden error response is returned to the client. Client IP: %s

error

Cause: The total length of the headers in the request exceeded the configured maximum.

Action: Adjust the maximum combined header size limit in the Security page of Oracle Enterprise Manager or OracleAS Web Cache Manager.

11357 Failed to resolve the requested site name. A bad host error response is returned to the client.

error

Cause: The request did not contain enough information about the site name.

Action: No action required.

11358 Failed client certificate check. A forbidden error response is returned.

error

Cause: The request was for a URL that requires a client certificate, and the certificate check failed for this request.

Action: No action required.

11359 Host header missing in HTTP/1.1 request. A bad host error response is returned.

error

Cause: The request was an HTTP/1.1 request, but is missing the required Host header.

Action: No action required.

11360 Error processing the Transfer-Encoding header in the request. A bad request error response is returned.

error

Cause: An error was encountered while processing the Transfer-Encoding header in the request.

Action: No action required.

11361 Transfer-Encoding specified in the request is not supported. An HTTP not implemented error response is returned.

error

Cause: OracleAS Web Cache only supports chunked encoding. Other types transfer encoding are not supported.

Action: No action required.

11362 Malformed request. A bad request error response is returned. Client IP: %s

error

Cause: The request was malformed.

Action: No action required.

11363 Out of memory while processing the request. The request is dropped.

error

Cause: OracleAS Web Cache failed to obtain enough memory while processing the request.

Action: Adjust the maximum cache size in the Resource Limits and Timeouts page of Oracle Enterprise Manager or Resource Limits page of OracleAS Web Cache Manager.

11364 Network error response is returned.

error

Cause: A network error was encountered while processing the request.

Action: Check other error messages related to this request.

11365 Server busy response is returned.

error

Cause: The origin servers for the requested site were at full capacity, and the waiting queue was full.

Action: Increase the capacity of the origin servers.

11366 A client connection to listening port %d is dropped.

warning

Cause: A connection failure was encountered between OracleAS Web Cache and the client, so the request could not be completed.

Action: No action required.

11367 No matching origin server can be found for the requested site. A bad host error response is returned

error

Cause: The requested site did not have any matching origin servers in the configuration.

Action: Modify the Sites page of Oracle Enterprise Manager or Site-to-Server Mapping page of OracleAS Web Cache Manager.

11368 ESI exception error response is returned.

error

Cause: An ESI exception was encountered while processing the request.

Action: Check other error messages related to this request.

11369 The request method is not supported. An HTTP not implemented error response is returned.

error

Cause: The request specified an unsupported method.

Action: No action required.

11370 A chunk-encoded request entity body contains a chunk that exceeds %u bytes. A bad request error response is returned. Client IP: %s

error

Cause: This chunk-encoded request contained a chunk that exceeded the DOCHEADERBUFFERSIZE setting in internal.xml

Action: Consider increasing DOCHEADERBUFFERSIZE.

11371 Malformed request. The chunk-encoded request entity body contains a line feed without a preceding carriage return. A bad request error response is returned. Client IP: %s

error

Cause: The request is malformed.

Action: none

11372 Aborting origin server request

warning

Cause: A request sent to OracleAS Web Cache was aborted. Because OracleAS Web Cache was in the process of forwarding the request to an origin server, the request to the origin server was also aborted.

Action: No action required.

11373 This request %s is not being enabled with end-user performance monitoring.

warning

Cause: A single response from the origin server to OracleAS Web Cache was not enabled for end-user performance monitoring because of a software limitation.

Action: No action required.

11376 The cached object is stale and this request will be served stale. However origin server is already being contacted for the fresh copy by some other request

trace

Cause: The cached object is stale and OracleAS Web Cache decided to serve the stale copy to the request. The origin server was already contacted for the fresh copy by some other request

Action: No action required.

11377 Aborting receiving request entity body because origin server connection has been aborted

trace

Cause: An error occurred while sending the request entity body to an origin server. The origin server connection was aborted. Because of this, OracleAS Web Cache is aborting the request received from the client.

Action: No action required.

11379 The request is using global site configuration.

trace

Cause: OracleAS Web Cache used global site configuration for this request.

Action: No action required.

11380 Request URL path prefix %s is re-written to %s based on site definition.

trace

Cause: URL path prefix was re-written based on the site configuration.

Action: No action required.

11381 A request is sent for a forbidden operation.

error

Cause: The request was attempted to be performed on an IP address for which the operation is forbidden.

Action: Check the IP address to ensure the correct target platform was specified.

11382 The URL for the statistics monitoring request is not valid.

error

Cause: The incorrect URL was used for the statistics monitoring request.

Action: Check the URL.

11385 The request fails because the service is unavailable.

error

Cause: The request was not processed because the service at the specified IP address was not available.

Action: Make the sure service is available prior to initiating the invalidation request.

11386 The request fails because the portal service is unavailable.

error

Cause: The request was not processed because the service at the specified IP address was not available.

Action: Make sure the service is available prior to initiating the invalidation request.

11387 The request fails due to a gateway timeout.

error

Cause: The request was not be processed due to a gateway timeout.

Action: Make sure the gateway is active and available for processing requests.

11389 The request fails due to unsupported ESI operation.

error

Cause: The request was not processed because no clients on the path of request can provide sufficient ESI processing capability.

Action: Check the offending ESI tag for correctness or possible typographical errors.

11390 The ECID %s overrides the temporary ECID %s.

notification

Cause: The ECID generated internally by OracleAS Web Cache was overridden by the new ECID discovered in the incoming Oracle-ECID request header. From now on, events logged within the request/response context will be displayed with the new ECID instead of the temporary ECID.

Action: No action required.

11391 Detailed information about ECID %s is in the next line.

notification

Cause: Additional information about the request represented by the ECID was acquired and will be printed in the next line in the event log.

Action: No action required.

11392 The object is in the cache, but the inactivity timeout cookie must be updated.

trace

Cause: The cached object will be returned to the client, but the inactivity timeout cookie must first be updated.

Action: No action required.

11393 The central cache or provider cache has challenged the remote or subscriber cache for identity information.

trace

Cause: The central cache or provider cache requested content from its parent cache. Before receiving content, it must register itself as an OracleAS Web Cache to establish a hierarchical relationship.

Action: No action required.

11394 The invalidation request is redundant.

trace

Cause: The invalidation request was not processed because the request was redundant with a previous requests. This error can occur in cache hierarchies.

Action: No action required.

11397 request failed due to insufficient memory

alert

Cause: The object request failed due to insufficient memory necessary to process all of the objects.

Action: Increase the size of the pagefile.

11398 A client connection to listening port %d is dropped at the end of keep-alive session.

trace

Cause: A connection between OracleAS Web Cache and the client was terminated, marking the end of the keep-alive session.

Action: No action required.

11399 The port number in the Host header %s is replaced with the port number of the connection %d.

trace

Cause: As a work around for an known Internet Explorer problem, the port number in the Host header was ignored and port number of the actual connection was used to determine the site instead.

Action: No action required.

11400 non-200 OK HTTP status code response code of %d

trace

Cause: The response received from the origin server contained a HTTP status code other than 200 OK.

Action: If this is not the intended response, then check the application code.

11403 begin cacheability decision for following url:

trace

Cause: This marks the start of the cacheability decision for the specified URL. All messages logged between this message and the "final cacheability decision made" message are about the same request/response pair.

Action: No action required.

11404 Time %s in the Last-Modified header is after the current time. The cache is using the current time instead.

warning

Cause: The Last-Modified response header received from the origin server contained a time that is in the future. The cache used the current time as the last-modified time instead.

Action: Check the application code that generates the Last-Modified response header, and make sure the machine where OracleAS Web Cache is running on has its clock in-sync with the clock of the application Web server.

11406 URL matches no-cache rule "%s".

trace

Cause: The request URL matched a non-cacheable caching rule.

Action: No action required.

11407 URL matches caching rule "%s".

trace

Cause: The request URL matched a cacheable caching rule.

Action: No action required.

11408 URL does not match any caching rule.

trace

Cause: The request URL did not match any configured caching rule.

Action: If this is not the intended behavior, then check the configured caching rules.

11409 HTTP POST request body length %d exceeds maximum cacheable limit. %d.

trace

Cause: The POST body length in the request exceeded the maximum cacheable limit for a POST body. Therefore, the corresponding response was not cached.

Action: If this response needs to be cached, contact Oracle Support.

11410 Response contains cookie "%s=%s", which is not present in the request.

trace

Cause: The response received from the origin server contained a newly created cookie that was not present in the request. As a result, OracleAS Web Cache considered the response to be non-cacheable.

Action: No action required.

11411 Response contains cookie "%s=%s", which does not match the same cookie value "%s" in the request.

trace

Cause: The response received from the origin server contained a new cookie that is different from the cookie value in the request. As a result, OracleAS Web Cache considered the response to be non-cacheable.

Action: No action required.

11412 The Surrogate-Control header conflicts with an existing caching rule. Cached objects associated with the previous rule will be marked for removal from the cache.

trace

Cause: The Surrogate-Control header in the response contained control directives that were in conflict with an existing caching rule. This will cause OracleAS Web Cache to mark all existing objects in the cache that were associated with the previous rule for removal.

Action: If this is not the intended behavior, then check the application logic, as well as the configured caching rule.

11413 The HTTP request method is not GET, or the URL contains query string parameters. The request is not cacheable.

trace

Cause: The request used an HTTP method that was neither GET nor POST. As a result, OracleAS Web Cache considered it to be non-cacheable.

Action: No action required.

11414 Initial cache key is composed:

trace

Cause: The initial cache key was composed of the complete site name, the URL, the HTTP request method, and the POST body, if any.

Action: No action required.

11415 Final cache key is composed:

trace

Cause: The final cache key was composed of the initial cache key plus any cookies and headers as specified by caching rule.

Action: No action required.

11416 The response is considered not cacheable due to the lack of caching directives.

trace

Cause: The response did not contain any headers with caching directives and had no matching caching rule. As a result, OracleAS Web Cache considered it to be non-cacheable.

Action: No action required.

11417 URL is not cacheable based on its HTTP headers.

trace

Cause: The response contained regular HTTP headers that indicated it was not cacheable. As a result, OracleAS Web Cache considered it to be non-cacheable.

Action: No action required.

11418 The response contains a Surrogate-Control header with either a no-store control directive or a zero-value max-age control directive, indicating that the response is non-cacheable.

trace

Cause: The response contained a Surrogate-Control header with either a no-store control directive or a zero-value max-age control directive. As a result, OracleAS Web Cache considered it to be non-cacheable.

Action: No action required.

11419 The Surrogate-Control header contains a nonzero max-age=%d+%d control directive, indicating that the response is cacheable.

trace

Cause: The response contained a Surrogate-Control header with a nonzero max-age directive. As a result, OracleAS Web Cache considered it to be cacheable.

Action: No action required.

11420 The response contains Surrogate-Control header with a zero-value max-age control directive and Etag header, indicating that the response is cacheable with validation.

trace

Cause: The response contained Surrogate-Control header with a zero-value max-age control directive and Etag header. As a result, OracleAS Web Cache considered it to be cacheable with validation.

Action: No action required.

11421 The response contains Etag and X-Oracle-Cache headers, indicating that the response is cacheable with validation.

trace

Cause: The response contained Etag and X-Oracle-Cache headers. As a result, OracleAS Web Cache considered it to be cacheable with validation.

Action: No action required.

11422 The Surrogate-Control header contains the vary control directive, overriding configuration.

trace

Cause: The response Surrogate-Control header contained the vary control directive. It is used in place of the existing caching rule.

Action: No action required.

11423 OracleAS Web Cache ran out of memory while creating caching rule according to the vary control directive of the Surrogate-Control header.

alert

Cause: There was not sufficient memory for OracleAS Web Cache to process the vary control directive.

Action: Adjust the maximum cache size in the Resource Limits and Timeouts page of Oracle Enterprise Manager or Resource Limits page of OracleAS Web Cache Manager.

11424 The request contains session %s, which makes it non-cacheable based on the existing caching rule.

trace

Cause: The current caching rule for this URL specified that if a specific session exists in the request, then the request is treated as non-cacheable.

Action: No action required.

11425 The request does not contain session %s, which makes it non-cacheable based on the existing caching rule.

trace

Cause: The current caching rule for this URL specified that if a specific session did not exist in the request, the request is treated as non-cacheable.

Action: No action required.

11426 The request does not contain cookie %s, which makes it non-cacheable based on the existing caching rule.

trace

Cause: The current caching rule for this URL specified that if a specific cookie does not exist in the request, the request is treated as non-cacheable.

Action: No action required.

11427 The request does not contain HTTP request header %s, which makes it non-cacheable based on the existing caching rule.

trace

Cause: The current caching rule for this URL specified that if a specific header does not exist in the request, the request was treated as non-cacheable.

Action: No action required.

11428 The X-Oracle-Expires header contains TTL of %d sec, indicating that the response is cacheable.

trace

Cause: The response contained an X-Oracle-Expires header with a nonzero TTL value. As a result, OracleAS Web Cache considered it to be cacheable.

Action: No action required.

11429 The response contains Etag header and zero value TTL in X-Oracle-Expires header, indicating that the response is cacheable with validation.

trace

Cause: The response contained an X-Oracle-Expires header with a zero value TTL and an Etag header. As a result, OracleAS Web Cache considered it to be cacheable with validation.

Action: No action required.

11430 The response contains zero value TTL in X-Oracle-Expires header with no Etag header, indicating that the response is non-cacheable.

trace

Cause: The response contained an X-Oracle-Expires header with a zero value TTL. As a result, OracleAS Web Cache considered it to be not cacheable.

Action: No action required.

11431 In absence of any overriding headers in the response, the response is not cacheable based on the existing caching rule.

trace

Cause: The response was not cacheable because the matching caching rule indicates so, and there are no overriding response headers in the response to override the caching rule.

Action: No action required.

11432 The HTTP response headers indicate that the response is cacheable. %s

trace

Cause: The response contained HTTP headers that indicate it is cacheable.

Action: No action required.

11433 An existing NCM entry states object is too large to be cacheable .

trace

Cause: The response body size was possibly greater than the maximum cacheable object size.

Action: If this is not the desired behavior, increase the maximum cached object size value in the Resource Limits and Timeouts page of Oracle Enterprise Manager or Resource Limits page of OracleAS Web Cache Manager.

11434 The ESI include or inline tag for this fragment contains reference TTL=%d+%d, making the fragment cacheable.

trace

Cause: The ESI include or inline tag for the fragment contained a nonzero reference TTL. As a result, OracleAS Web Cache considered the fragment cacheable.

Action: No action required.

11442 The response could not be parsed.

trace

Cause: The Surrogate-Control response header contained a content type but a Content header is not text or HTML.

Action: If the content needs to be parsed, send the text or HTML in the Content header.

11443 The request does not contain header %s, which makes it non-cacheable based on the vary control directive in the Surrogate-Control response header.

trace

Cause: The request was missing a required header to be cacheable based on the vary control directive in the Surrogate-Control response header.

Action: No action required.

11444 The request does not contain cookie %s, which makes it non-cacheable based on the vary control directive in the Surrogate-Control response header.

trace

Cause: The request was missing a required cookie to be cacheable based on the vary control directive in the Surrogate-Control response header.

Action: No action required.

11445 Following URL will not be cached:

trace

Cause: The final cacheability decision for this response is not cacheable.

Action: No action required.

11446 URL which will be cached is:

trace

Cause: The final cacheability decision for this response is cacheable.

Action: No action required.

11447 Request URL matches a dynamic caching rule.

trace

Cause: The request URL matched a dynamic caching rule.

Action: No action required caching rules.

11448 URL matches disabled caching rule "%s".

trace

Cause: The request URL matched a disabled caching rule; thus, the rule is ignored.

Action: No action required.

11449 The response is too large to be cached.

trace

Cause: The response size is larger than the configured maximum   size of a single cached object.

Action: Check the configured Maximum Size of a Single Cached Object in the Resource Limits and Timeouts page of Oracle Enterprise Manager or the Maximum Cached Object Size in the Resource Limits page of OracleAS Web Cache Manager.

11450 Object is cached because it does not exceed allowed size for an object.

trace

Cause: The object was cacheable because its response was within the configured maximum object size. Previously, the object was uncacheable because it exceeded the limit.

Action: No action required.

11451 Update fails because object is no longer cacheable.

trace

Cause: This request was no longer cacheable.

Action: No action required.

11452 Update fails; OracleAS Web Cache fetches response as a cache miss.

trace

Cause: An update of the object failed. OracleAS Web Cache processes request as a cache miss

Action: No action required.

11453 A new cache policy is created based on the Surrogate-Control header.

trace

Cause: The caching properties specified in the Surrogate-Control header conflicts with the existing cache policy, therefore OracleAS Web Cache created a new policy to be used for future requests.

Action: If this is not the intended behavior, then check the application logic, as well as the configured caching rule.

11454 The compression setting derived from the Surrogate-Control header or the absence of the header conflicts with the existing setting. Cached objects associated with the previous setting will be marked for removal from the cache.

trace

Cause: The response contains a compression setting either explicitly through the Surrogate-Control header or implicitly through the absence of it that was in conflict with an existing compression setting. This will cause OracleAS Web Cache to mark all existing objects in the cache that were associated with the previous setting for removal.

Action: If this is not the intended behavior, then check the application logic, as well as the configured caching rule.

11501 While getting detailed memory statistics, pointer swapped while filling HTML

notification

Cause: While retrieving detailed memory statistics, the timer pointer swapped to a new index as it was filling the HTML template.

Action: No action required.

11502 Unable to create or open file %s for write. Cache contents not written.

error

Cause: The cache server was unable to open the file specified. Either the file already exists and the OracleAS Web Cache user does not have write permission on the file or the OracleAS Web Cache user does not have write permission to the directory.

Action: Check permissions on the file and the directory and correct them to allow OracleAS Web Cache to create the file or write to it.

11503 Abnormal statistics value of %s was detected. Value 0 was recorded.

alert

Cause: The statistics value that was returned was an negative value. This problem is typically caused by a OracleAS Web Cache server that has been active for a long period of time. OracleAS Web Cache automatically reset the value.

Action: No action required.

11504 Client IP checking for the statistics message fails. Access denied.

error

Cause: OracleAS Web Cache allows statistics requests to be sent from configured trusted hosts or trusted subnets. The current statistics request message was not sent from an IP address that belongs to a trusted host or subnet.

Action: Add the IP address for the computer as trusted host in the Security page of Oracle Enterprise Manager or the Security page of OracleAS Web Cache Manager.

11505 The number of statistics monitoring requests exceeds queue limit.

alert

Cause: The number of statistics monitoring requests exceeded the request queue limit.

Action: Try the request at a later time.

11700 Client IP checking for the invalidation message fails. Access denied.

error

Cause: OracleAS Web Cache allows invalidation to be sent from configured IP groups. The current invalidation message was not sent from an IP address that belongs to the configured groups.

Action: Configure trusted hosts in the Security page of Oracle Enterprise Manager or OracleAS Web Cache Manager.

11701 Authentication failed. Invalid username or password.

error

Cause: The invalidation message contained an incorrect username or password.

Action: Resend the invalidation message with the correct invalidator user name and password. You can change the password in the Security page of Oracle Enterprise Manager or OracleAS Web Cache Manager.

11702 The invalidation request contains an empty request body.

error

Cause: The invalidation request contained an empty request body.

Action: Resend the invalidation request with a request body.

11703 Invalidation request specifies a wrong URL.

error

Cause: The invalidation message carried a wrong request URL.

Action: No action required.

11704 Invalidation message contains XML parsing errors.

error

Cause: The invalidation message contained invalid XML.

Action: Correct the XML errors in the invalidation message.

11705 Invalid validity level '%s' is requested.

error

Cause: The invalidation message specified an invalid validity level. The correct range of level is between 0 and 9.

Action: Correct the message to use a validity level between 0 to 9.

11706 %d object(s) matching prefix '%s' are invalidated.

notification

Cause: Invalidation of objects was successful.

Action: No action required.

11707 Object with URL '%s' is successfully invalidated.

notification

Cause: Invalidation was successful.

Action: No action required.

11708 Object with URL '%s' is not cacheable. Invalidation is not done.

notification

Cause: The URL was not a cacheable object. Invalidation is not performed for non-cacheable objects.

Action: No action required.

11710 Object with URL '%s' not in the cache. Invalidation is not performed.

notification

Cause: The object was not in the cache. Invalidation is not performed for non-cacheable objects.

Action: No action required.

11711 OracleAS Web Cache cannot compose key pattern for requested URL '%s'.

error

Cause: OracleAS Web Cache could not compose a key pattern for requested URL expression.

Action: Correct the URL expression in the invalidation message.

11712 Invalidation message contains unrecognized cookies.

error

Cause: The invalidation message contained nonexistent cookies.

Action: Correct the cookie name specified in the invalidation message.

11713 Invalid regular expression found in invalidation message. %s

error

Cause: The invalidation message contained an invalid regular expression that could not be compiled.

Action: Correct the regular expression in the invalidation message.

11715 Regular expression matching error. Error number: %d

error

Cause: The regular expression in the invalidation message was invalid. OracleAS Web Cache could not perform regular expression matching successfully.

Action: Check regexec man page with the error number for syntax error description.

11716 Invalidation XML message cannot be parsed.

error

Cause: Invalidation XML message could not be parsed.

Action: No action required.

11717 Invalidation XML error found in invalidation message.

error

Cause: OracleAS Web Cache received an invalidation message in an old version. It is unable to obtain all information about the URL node.

Action: Correct the invalidation message or use an invalidation version of either "WCS-1.1" or "WCS-1.0".

11721 Cannot read value for XML element/attribute: '%s'.

error

Cause: OracleAS Web Cache was unable to obtain all information about the specified XML element or attribute.

Action: Correct the XML in the invalidation message.

11722 Invalidation message specifies unknown version: '%s'.

error

Cause: The invalidation message specified an unknown version.

Action: Correct the invalidation message to use version "WCS-1.1" or "WCS-1.0".

11723 Value of XML string '%s' too long. Actual length: %d, max length allowed: %d.

error

Cause: The XML string in the invalidation message exceeded the allowed length limit.

Action: Correct the XML in the invalidation message.

11724 URL path prefix does not end with '/': '%s'.

error

Cause: The URL path prefix did not end with '/' in the invalidation message.

Action: Correct the syntax of the URL path prefix in the invalidation message.

11725 Cannot create XML element/attribute: '%s'.

error

Cause: OracleAS Web Cache could not create the XML element or attribute.

Action: Check the maximum cache size in the Resource Limits and Timeouts page of Oracle Enterprise Manager or Resource Limits page of OracleAS Web Cache Manager. Restart OracleAS Web Cache if necessary.

11726 Cannot set XML attribute '%s' for element '%s'.

error

Cause: OracleAS Web Cache could not set the XML attribute value.

Action: Check the maximum cache size in the Resource Limits and Timeouts page of Oracle Enterprise Manager or Resource Limits page of OracleAS Web Cache Manager. Restart OracleAS Web Cache if necessary.

11727 Unexpected/unsupported argument '%s' in function '%s'.

error

Cause: The invalidation message specified an unsupported name and value pair for <SYSTEMINFO ... />

Action: See chapter "Sending Invalidation Requests" in the OracleAS Web Cache Administrator's Guide for the possible values allowed for <SYSTEMINFO ... />.

11728 Cluster error: Peer cache '%d' returns bad response.

error

Cause: The invalidation response from a peer cache was not parsed.

Action: Check the network connection between current cache and its peer.

11729 Invalidation message contains conflicting HOST values: '%s' and '%s'.

warning

Cause: The invalidation message contained conflicting HOST values in the URIPREFIX and the HOST attributes.

Action: Correct the host information in the invalidation message.

11730 Invalidation message from provider cache with IP '%s' port '%d' is dropped.

warning

Cause: The invalidation message propagated from the specified provider cache is either outdated or invalid. Invalidation is discarded.

Action: No action required.

11731 Normal request is from an unknown OracleAS Web Cache: IP '%s' port '%s'.

warning

Cause: The invalidation message was propagated from an unknown cache. Invalidation is discarded.

Action: No action required.

11732 Invalidation contains host name '%s' which does not match the IP '%s' from which invalidation comes.

warning

Cause: The invalidation message propagated from a cache provided no host name or the host name in the URIPREFIX or HOST attribute was not allowed for that cache. Invalidation is discarded.

Action: Correct the host information in the invalidation message.

11733 Invalidation object is ignored due to lack of host information.

notification

Cause: Invalidation propagated from a cache does not specify a host name in the URIPREFIX or HOST attribute of the message.

Action: Correct the host information in the invalidation message.

11734 Invalidation sent to subscriber cache with IP '%s' port '%d' has returned with response code: '%s'.

notification

Cause: The provider cache response was received.

Action: No action required.

11735 Subscriber cache with IP '%s' port '%d' is removed due to an exceeded failure count. Total subscriber(s): '%d'.

notification

Cause: Subscriber cache was removed from the internal subscriber list because the invalidation propagated to it has failed too many times.

Action: No action required.

11737 Invalidation message may be lost from provider cache with IP '%s' port '%d'.

warning

Cause: The subscriber cache missed an invalidation message from the provider cache. All objects are invalidated.

Action: No action required.

11738 Provider cache fails in propagating invalidation message to subscriber cache.

error

Cause: The provider cache failed in propagating the invalidation message to subscriber cache.

Action: Check the configuration file. Restart OracleAS Web Cache if necessary.

11739 New subscriber OracleAS Web Cache with IP '%s' port '%d' has been established. Total subscriber(s): '%d'.

notification

Cause: The new subscriber cache was added to the internal subscriber list.

Action: No action required.

11740 Invalidation message contains too many cookies, headers, or attributes. Maximum allowed is '%d'.

error

Cause: Invalidation message contained too many cookies, headers, or attributes. Invalidation was not performed.

Action: Rephrase the invalidation message.

11741 Invalidation message received from OracleAS Web Cache with IP '%s' port '%d', sequence number current: '%d', passed: '%d'.

notification

Cause: OracleAS Web Cache received a propagated invalidation message.

Action: No action required.

11742 Invalidation sent from IP address '%s' contains host name '%s', is not allowed from that IP address%s

warning

Cause: The propagated invalidation message contained an invalid host name in the URIPREFIX or HOST attribute.

Action: Correct the host information in the original invalidation message.

11743 Asynchronous mode is disabled.

notification

Cause: OracleAS Web Cache is set not to chunk invalidation processing.

Action: No action required.

11744 Asynchronous mode is enabled; chunk time is %d milliseconds.

notification

Cause: OracleAS Web Cache is set to chunk invalidation processing.

Action: No action required.

11747 Running in "disconnected" mode

trace

Cause: The <SYSTEMINFO ... /> element of the invalidation message was configured with NAME="WCS_DISCONNECTED_OK VALUE="YES". As a result, the invalidation was performed without waiting for an invalidation response for each invalidation message.

Action: No action required.

11748 Invalidation with INFO '%s' has returned with status '%s'; number of objects invalidated: '%d'.

notification

Cause: The result of the invalidation was successful. INFO is the comment specified in the INFO element of the invalidation request. The status can be SUCCESS for successful invalidations, URI NOT CACHEABLE for objects that are not cacheable, or URI NOT FOUND for objects not found in the cache.

Action: No action required.

11750 Invalid syntax for the value of QUERYSTRING_PARAMETER in invalidation message:'%s'.

error

Cause: The invalidation message contained invalid syntax in the QUERYSTRING_PARAMETER value.

Action: Correct the error in invalidation message.

11751 Subscriber cache not inserted into internal subscriber list for itself.

warning

Cause: The subscriber cache was not inserted into the internal subscriber list for itself. The list for the cache could not contain entries for its own cache.

Action: Check the cache topology to verify that loopback is intended.

11752 Cannot decode %s with value '%s', therefore decoding is not done.

warning

Cause: Cannot decode XML value specified. The original value is used.

Action: No action required.

11753 Invalid host name '%s' specified in invalidation message %s

error

Cause: Invalid host name was specified in the URIPREFIX or HOST attribute of the invalidation message.

Action: Correct the host information in the invalidation message.

11755 Default URL size is too small for cache key.

error

Cause: Default URL size was too small for cache key.

Action: No action required.

11757 Unsupported OTHER element specified in invalidation message: name='%s', type='%s', value='%s'.

warning

Cause: Unsupported OTHER element was specified in the invalidation message.

Action: See chapter "Sending Invalidation Requests" in the OracleAS Web Cache Administrator's Guide for all supported OTHER elements.

11800 CGI communication error.

Cause: Network I/O error while communicating with CGI application.

Action: No action required.

11802 Cannot open Service Manager: %s

alert

Cause: The admin server was unable open the Windows Service Manager.

Action: If the problem persists, contact Oracle Support.

11803 Cannot open Windows Service %s: %s

alert

Cause: The admin server was unable to open the specified service.

Action: If the problem persists, contact Oracle Support.

11804 Cannot obtain the system status for Windows service %s: %s

alert

Cause: The admin server was unable to retrieve the status for the specified service.

Action: If the problem persists, contact Oracle Support.

11805 Cannot control service %s: %s

alert

Cause: The admin server was unable to start or stop the specified Windows service.

Action: If the problem persists, contact Oracle Support.

11806 Cannot stop service %s

alert

Cause: The admin server failed to stop the specified service.

Action: If the problem persists, contact Oracle Support.

11807 Windows service, %s, stopped

notification

Cause: The admin server has successfully stopped the specified service.

Action: No action required.

11809 The admin server could not start the cache server, running in admin-only mode: %s

alert

Cause: The admin server was unable to start the cache server.

Action: Review the error log file or use the Event Viewer to help determine the cause of the problem.

11810 Cannot start Auto-Restart. %s does not exist

alert

Cause: Unable to find the specified Auto-Restart executable.

Action: Restore the Auto-Restart executable, webcachemon, to its proper location, which is the OracleAS Web Cache bin directory.

11811 Unable to fork a new process for starting Auto-Restart.

alert

Cause: OracleAS Web Cache was unable to fork a new process.

Action: This is probably a temporary system resource issue. If the problem persists, contact Oracle Support.

11812 Auto-Restart did not start. An error code was returned.

alert

Cause: The Auto-Restart process failed to start.

Action: There may be associated error messages from the Auto-Restart process indicating why it was unable to start. If there are no other error messages, it may be a system resource problem. For example, the exec() call may have failed. If the problem persists, contact Oracle Support.

11813 Cannot start service %s

alert

Cause: The admin server failed to start the specified service.

Action: If the problem persists, contact Oracle Support.

11814 Logging info: %s

error

Cause: An error is logged.

Action: Refer to the logged error details and take the appropriate action.

11815 The admin server process is unable to determine the OPMN auto-restart setting. Error reading or parsing %s

warning

Cause: An error occurred while the admin server process was attempting to obtain the restart setting for OracleAS Web Cache in the OPMN XML configuration file, opmn.xml.

Action: If the problem persists, contact Oracle Support.

11816 Error while changing the OPMN auto-restart setting for OracleAS Web Cache. Command: %s. Return code: %d. Command output: %s

warning

Cause: An error occurred while trying to execute the command specified. OracleAS Web Cache uses this command to change the restart setting in the OPMN XML configuration file, opmn.xml.

Action: Execute the command manually.

11817 Unable to allocate or access a shared memory segment of size %u bytes. %s(): %s

alert

Cause: The specified operating-system level function, which creates or accesses a shared memory segment of the size specified in the error message, has failed for the specified reason.

Action: This error usually occurs as a result of insufficient system resources. Refer to the included operating system level error message for further information.

11818 Login attempt fails to admin server process.

warning

Cause: An incorrect username or password was entered while attempting to access the admin server process. If this error occurs frequently, a malicious user might be attempting to crack the password

Action: Try the username and password again.

11901 SSL context creation failed, NZE-%d

alert

Cause: SSL library failed to create a context.

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

11902 SSL configuration fails, NZE-%d

warning

Cause: SSL library configuration failed.

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

11903 SSL set credential fails, NZE-%d

warning

Cause: SSL library failed to set credentials.

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

11904 SSL handshake fails NZE-%d

warning

Cause: SSL handshake error occurred.

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

11907 The server cannot allocate memory while in %s.

alert

Cause: The system has run out of memory.

Action: Restart the system and OracleAS Web Cache.

11908 SSL set peerid fails, NZE-%d

warning

Cause: SSL library failed to create SSL Session

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

11909 OracleAS Web Cache fails to get the name of communicating peer during SSL handshake, system error %d.

warning

Cause: OracleAS Web Cache could not determine peer IP Address.

Action: Contact Oracle Support.

11910 failed to normalize Wallet Resource Locator (WRL) %s

warning

Cause: WRL could not be parsed.

Action: Confirm that the WRL is in the correct format.

11913 SSL set hardware acceleration fails, NZE-%d.

alert

Cause: The SSL library failed to initialize hardware SSL acceleration.

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

11914 SSL set thread usage fails, NZE-%d.

warning

Cause: The SSL library failed to set threading model.

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

11915 The SSL engine fails to initialize, NZE-%d.

warning

Cause: The SSL library engine failed to initialize.

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

11916 Oracle cryptographic toolkit context initialization fails, NZE-%d.

warning

Cause: The SSL library toolkit context failed to initialize.

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

11917 SSL wallet %s file %s does not exist.

warning

Cause: The wallet was not found in the wallet directory.

Action: Check for wallet file ewallet.p12 or ewallet.der in the wallet directory.

11918 The SSL Wallet %s file %s is not readable.

warning

Cause: The wallet was not readable by OracleAS Web Cache.

Action: Check the file system permissions on this file.

11919 The SSL wallet autologin file %s does not exist. Wallet does not appear to be autologin wallet.

warning

Cause: The wallet autologin file cwallet.sso was not found.

Action: Check for wallet file cwallet.sso in wallet directory.

11920 The SSL wallet autologin file %s is not readable. Wallet does not appear to be autologin wallet.

warning

Cause: The wallet autologin file cwallet.sso was not readable by OracleAS Web Cache.

Action: Check the file system permissions on this file.

11921 The origin server wallet did not open. Operating without wallet for backend. Only Diffie-Hellman anonymous connections supported to origin servers.

warning

Cause: Origin server wallet did not open or was not specified.

Action: Without an origin server wallet, only Diffie-Hellman SSL Connections can be made.

11922 %s wallet fails to open at location %s, NZE-%d, as user %s

warning

Cause: The wallet could not be opened as the user specified.

Action: Check for the existence of the wallet and make sure autologin was enables as the user specified.

11923 %s wallet fails to open at location %s, NZE-%d.

warning

Cause: The wallet could not be opened

Action: Check for the existence of the wallet.

11924 Oracle wallet fails to open persona, NZE-%d.

warning

Cause: The SSL library failed to retrieve certificate from wallet.

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

11925 No wallet configured with SSL; server initialization fails.

alert

Cause: SSL was configured without specifying a wallet.

Action: Correct the configuration file.

11926 OracleAS Web Cache fails to set cipher suite in SSL handshake, NZE-%d.

warning

Cause: The SSL library failed to set the cipher suite.

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

11927 OracleAS Web Cache fails to get supported cipher suites, NZE-%d.

alert

Cause: The SSL library failed to get the supported cipher suites.

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

11928 OracleAS Web Cache fails to set session reference, NZE-%d.

alert

Cause: The SSL library failed to get the supported cipher suites.

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

11929 SSL handshake fails NZE-%s.

warning

Cause: SSL handshake error occurred.

Action: See the "Network Security Messages (NZE)" chapter of the Oracle Database Error Messages for information about NZE errors.

11950 Internal XML parser error. Error code: %d. Error Message: %s

error

Cause: An internal error occurred while inside the XML parser. OracleAS Web Cache will attempt to log that internal error.

Action: This error usually occurs due to malformed XML. Look up the associated error code in the Oracle XDK documentation.

11951 invalid value for ESI redirect attribute

warning

Cause: The value specified for the redirect attribute by the <esi:include> tag was invalid.

Action: Set the value of the redirect attribute value to either "yes" or "no" in the ESI template.

11952 Start processing ESI document %s%s, nesting level %d

trace

Cause: Start processing of an ESI document (template or fragment).

Action: No action required.

11953 In ESI template %s%s, the fragment's site name and URL has been discovered as %s and %s

trace

Cause: Start processing of an ESI fragment.

Action: No action required.

11960 OracleAS Web Cache is unable to decode the URI passed in: %s

warning

Cause: URI decoding was unsuccessful, and it will be used as it is.

Action: No action required.

11962 OracleAS Web Cache is unable to allocate enough memory to parse ESI vars tag string.

error

Cause: There was not sufficient memory for OracleAS Web Cache to parse the string in the ESI vars tag.

Action: Adjust the maximum cache size in the Resource Limits and Timeouts page of Oracle Enterprise Manager or Resource Limits page of OracleAS Web Cache Manager.

11963 OracleAS Web Cache failed to create new ESI fragment request. Template uri: %s

error

Cause: ESI fragment request could not be generated. This error is generally a result of insufficient memory.

Action: Adjust the maximum cache size in the Resource Limits and Timeouts page of Oracle Enterprise Manager or Resource Limits page of OracleAS Web Cache Manager.

11964 empty variable key in ESI vars tag

error

Cause: The variable key specified in the ESI template by the ESI vars tag was empty. "key" was missing from $(VARIABLE_NAME{key="value"}).

Action: Correct the format of the ESI vars tag.

11965 invalid variable name in ESI vars tag

error

Cause: The variable name specified in the ESI template by the ESI vars tag is not defined.

Action: See appendix "Edge Side Includes (ESI) Language Tags" in the OracleAS Web Cache Administrator's Guide for valid ESI variable names.

11966 In %s%s, request header failed to be added to the include request.

error

Cause: There was not sufficient memory for OracleAS Web Cache to include the request, or there was a request header parsing error.

Action: Adjust the maximum cache size in the Resource Limits and Timeouts page of Oracle Enterprise Manager or Resource Limits page of OracleAS Web Cache Manager.

11973 Site name cannot be resolved.

error

Cause: Requested site in ESI fragment request could not be resolved.

Action: Make sure that site name in the ESI request is correct.

11974 Incomplete URI for ESI fragment.

error

Cause: URI for ESI fragment only had "http://" and was considered incomplete.

Action: Make sure that the URL in the ESI request is complete.

11975 Corrupted data in ESI environment metadata block.

error

Cause: The application delivered data that could not be parsed to OracleAS Web Cache.

Action: Fix the application.

11976 Duplicate name %s in ESI environment metadata block.

trace

Cause: The environment block contained multiple name entries.

Action: Remove the duplicate entries. Only the first entry is used.

11977 Memory allocation failure while initializing login metadata hash table.

alert

Cause: The OracleAS Web Cache memory manager could not allocate any more memory.

Action: Adjust the maximum cache size in the Resource Limits and Timeouts page of Oracle Enterprise Manager or Resource Limits page of OracleAS Web Cache Manager.

11978 In ESI template %s%s, failed to add request header value.

error

Cause: Wrong request-header was specified.

Action: Check the ESI template file and correct the error in request_header attribute.

11979 In ESI template %s%s, failed to add the request header because value is missing

error

Cause: Value field in the request-header was missing.

Action: Check the ESI template file and correct the error in the request_header attribute

11984 OracleAS Web Cache is unable to find the directory mapping to the default ESI fragment page %s.

alert

Cause: The default ESI fragment was not readable or did not exist in the $ORACLE_HOME/webcache/docs on UNIX or ORACLE_HOME\webcache\docs directory on Windows.

Action: Create a readable default ESI fragment in the $ORACLE_HOME/webcache/docs on UNIX or ORACLE_HOME\webcache\docs directory on Windows, and then configure it in the Sites page or Oracle Enterprise Manager or the Error Pages of OracleAS Web Cache Manager.

11985 OracleAS Web Cache is unable to obtain the size of the default ESI fragment page %s.

alert

Cause: The default ESI fragment was not readable or did not exist in the $ORACLE_HOME/webcache/docs on UNIX or ORACLE_HOME\webcache\docs directory on Windows.

Action: Create a readable default ESI fragment in the $ORACLE_HOME/webcache/docs on UNIX or ORACLE_HOME\webcache\docs directory on Windows, and then configure it in the Error Pages of OracleAS Web Cache Manager.

11986 The default ESI fragment page %s is too large.

alert

Cause: The default ESI fragment was too large.

Action: Alter the size of the default ESI fragment to less than 32 KB.

11987 OracleAS Web Cache fails to open the default ESI fragment page %s.

alert

Cause: The default ESI fragment was not readable or did not exist in the $ORACLE_HOME/webcache/docs on UNIX or ORACLE_HOME\webcache\docs directory on Windows.

Action: Create a readable default ESI fragment in the $ORACLE_HOME/webcache/docs on UNIX or ORACLE_HOME\webcache\docs directory on Windows, and then configure it in the Error Pages of OracleAS Web Cache Manager.

11989 read error while loading default ESI fragment page %s

alert

Cause: The default ESI fragment was not readable or did not exist in the $ORACLE_HOME/webcache/docs on UNIX or ORACLE_HOME\webcache\docs directory on Windows.

Action: Create a readable default ESI fragment in the $ORACLE_HOME/webcache/docs on UNIX or ORACLE_HOME\webcache\docs directory on Windows, and then configure it in the Error Pages of OracleAS Web Cache Manager.

11997 ESI stack overflow

error

Cause: Current stack maximum depth was set too low.

Action: Contact Oracle Support about how to increase the maximum stack depth.

12001 ESI <esi:include> tag nesting in template %s%s is too deep.

warning

Cause: The fragment <esi:include> depth exceeded the maximum include depth of three levels.

Action: Decrease the depth number in the ESI template, or contact Oracle Support to change the maximum <esi:include> depth.

12002 Origin server busy exception in ESI template %s%s, fragment %s%s

warning

Cause: The origin server was busy and could not accept the new request.

Action: Check the health status of the origin server. If needed, increase the origin server capacity in the Origin Servers page of Oracle Enterprise Manager or OracleAS Web Cache Manager.

12003 No connection exception in ESI template %s%s, fragment %s%s

warning

Cause: The cache was unable to connect to the origin server.

Action: Check the health status of the origin server. If needed, increase the origin server capacity in the Origin Servers page of Oracle Enterprise Manager or OracleAS Web Cache Manager.

12004 Network timeout exception in ESI template %s%s, fragment %s%s

warning

Cause: The request to origin server timed out during a network connection.

Action: Check the health status of the origin server. If needed, increase the origin server capacity in the Origin Servers page of Oracle Enterprise Manager or OracleAS Web Cache Manager.

12005 HTTP Client Error 4xx exception in ESI template %s%s, fragment %s%s
warning
Cause: The origin server returned an HTTP 4xx status code.

Action: Check for errors in the ESI template and fragment files.

12006 HTTP Server Error 5xx exception in ESI template %s%s, fragment %s%s
warning
Cause: The origin server returned an HTTP 5xx status code.

Action: Check for errors in the ESI template and fragment files. Additionally, check the health status of the origin server.

12007 Incompatible fragment version exception in ESI template %s%s, fragment %s%s
warning
Cause: The value for the content control directive in the Surrogate-Control response-header was not compatible between the ESI fragment and template.

Action: Correct the error in either the ESI template or fragment file.

12008 Incorrect response header exception in ESI template %s%s, fragment %s%s
warning
Cause: The response header of the ESI fragment caused an error.

Action: Correct the error in the ESI fragment.

12009 Incorrect ESI fragment exception in ESI template %s%s, fragment %s%s
warning
Cause: OracleAS Web Cache was unable to parse the ESI fragment.

Action: Correct the error in the ESI fragment.

12010 Incorrect XML Fragment exception in ESI template %s%s, fragment %s%s
warning
Cause: OracleAS Web Cache was unable to parse the XML fragment.

Action: Correct the error in the XML fragment.

12011 Generic exception in ESI template %s%s, fragment %s%s
warning
Cause: A generic ESI exception occurred.

Action: Check the template and fragment, and then correct the error.

12012 No exception handler is defined in template %s%s:.
error
Cause: An uncaught ESI exception was detected. An error page is returned.
Action: Correct the syntax error in the ESI template.

12013 No resolved virtual host mapping was found for template.
trace
Cause: A ESI template contained an unresolved virtual host mapping.

Action: No action required.

12015 Non-fetchable inline fragment %s%s not found in cache or already expired, fetching template %s%s from origin server again.
trace

Cause: Normal trace behavior.

Action: No action required.

12081 ESI parsing error. Dumping ESI content:

error

Cause: ESI parsing error.

Action: For any syntax errors, correct the ESI syntax. For any non-syntax related error, contact Oracle Support Services.

12082 OracleAS Web Cache cannot parse ESI due to insufficient memory.

error

Cause: OracleAS Web Cache ran out of memory while parsing ESI.

Action: Adjust the maximum cache size in the Resource Limits and Timeouts page of Oracle Enterprise Manager or Resource Limits page of OracleAS Web Cache Manager.

12083 ESI parser syntax error at line %d: %s

error

Cause: An ESI tag closed too early or there was a syntax error of the ESI parser.

Action: Correct the ESI syntax.

12085 ESI syntax error. Bad exception type at line %d.

error

Cause: An invalid token was encountered.

Action: Correct the error at the specified location in the ESI template.

12086 ESI syntax error. Unrecognized keyword %s is at line %d.

error

Cause: An invalid token was encountered.

Action: Correct the error at the specified location in the ESI template.

12087 Bad Location response header in ESI fragment response

error

Cause: A redirection response contained an invalid Location header.

Action: Make sure that the application Web server sends a valid Location header.

12088 ESI syntax error. The src attribute is missing at line %d.

error

Cause: The src attribute is required for the <esi:include> and <esi:environment> tags.

Action: Correct the error at the specified location in the ESI template.

12089 Syntax error. The name attribute is missing at line %d.

error

Cause: The name attribute is required for the <esi:inline> tag, <esi:environment> tag, and the request_header element.

Action: Include the name attribute at the specified location in the ESI template.

12090 ESI syntax error. Bad attribute value appears at line %d.

error

Cause: An invalid or empty value string was found.

Action: Correct the error at the specified location in the ESI template.

12091 ESI syntax error. Multiple ESI <esi:environment> tags appear in a single template at line %d

error

Cause: Multiple <esi:environment> tags in a single ESI template are not allowed.

Action: Configure the ESI template with one <esi:environment> tag.

12092 ESI Syntax error. Multiple request_body elements appear at line %d.

error

Cause: Multiple request_body elements are not allowed in <esi:include> and <esi:environment> tags.

Action: Correct the error at the specified location in the ESI template.

12093 ESI syntax error. The max-age attribute is incorrectly formatted at line %d.

error

Cause: The max-age attribute is not in the format of: max-age="expiration_time[+removal_time]"

Action: Correct the error at the specified location in the ESI template.

12094 ESI syntax error. String length exceeds maximum at line %d.

error

Cause: The string length exceeded the maximum limit.

Action: Contact Oracle Support to change the string length limit.

12095 ESI syntax error. A variable requires a key at line %d.

error

Cause: The specified variable requires a format of $(VARIABLE_NAME{key}).

Action: Include a key in the variable at the specified location in the ESI template.

12096 ESI syntax error. The POST request method requires an request_body at line %d.

error

Cause: The specified POST fragment did not have a request_body element.

Action: Include the request_body element at the specified location in the ESI template.

12097 ESI syntax error. The position of the <esi:environment> tag at line %d is invalid.

error

Cause: The <esi:environment> did not appear before any <esi:include> tags in the syntax. To use the custom variables specified by the <esi:environment> tag with other ESI tags, the <esi:environment> must be included before other <esi:include> tags.

Action: Correct the error at the specified location in the ESI template.

12098 ESI syntax error. The value of the fetchable attribute at line %d is invalid.

error

Cause: A value of yes or no are the only possible values for the fetchable attribute.

Action: Change the value to either yes or no at the specified location in the ESI template.

12099 ESI syntax error. The same attribute appears multiple times in an ESI element at line %d.

error

Cause: An attribute can appear only once within an ESI element.

Action: Correct the error at the specified location in ESI object.

12100 Exhausted process memory. Exiting process.

alert

Cause: Memory usage is greater than the process' system memory allocation.

Action: Reduce the maximum cache size limit or increase the operating system's memory limit for the OracleAS Web Cache process, if possible.

12101 Maximum receive buffer size 2 MB reached.

error

Cause: Receive buffer size is bigger than 2 MB on Windows platforms.

Action: Reduce receive buffer size to below 2 MB.

12102 Unable to locate memory block (%db). OracleAS Web Cache extends cache size over maximum limit to %dMB.

alert

Cause: Memory request was not fulfilled with existing process memory, even after forced garbage collection.

Action: If the problem persists, contact Oracle Support.

12103 Unable to locate large memory block (%db). OracleAS Web Cache extends cache size over maximum limit to %dMB.

alert

Cause: Large block memory request was not satisfied with existing process memory, even after forced garbage collection.

Action: If the problem persists, contact Oracle Support.

12201 Cache memory allocation for the cluster member table failed.

error

Cause: Memory allocation failed.

Action: No action required.

12202 Cluster member DNS lookup problem

trace

Cause: Could not resolve the IP address from the cluster member's host name.

Action: Check the Cluster Members and Properties page of Oracle Enterprise Manager or the Clustering page of OracleAS Web Cache Manager to see if the host name is correctly listed.

12203 Cluster member number %s owner slot allocation %s:%s

trace

Cause: Normal trace behavior.

Action: No action required.

12204 Add cluster member number %s

notification

Cause: Normal notification behavior.

Action: No action required.

12205 Remove cluster member number %s

notification

Cause: Normal notification behavior.

Action: No action required.

12206 Cluster member number %s already marked alive

warning

Cause: The local cache tried to add the specified cluster member, but it was already marked as alive.

Action: No action required.

12207 Cluster member number %s already marked dead

warning

Cause: The local cache tried to remove the specified cluster member, but it was already marked as dead.

Action: No action required.

12209 A %s node cluster successfully initialized

notification

Cause: Normal notification behavior.

Action: No action required.

12212 Cluster member %s - %s configuration is invalid.

warning

Cause: The configuration file of the specified cluster member does not match the configuration of the local cache.

Action: Check that all cluster members have the same configuration file.

12213 Cluster member %s - %s configuration is valid.

notification

Cause: Normal notification behavior.

Action: No action required.

12217 Cluster member count %s is larger than maximum allowed %s

error

Cause: There are more than the maximum number of cluster members. There cannot be more than 99 cluster members.

Action: Reduce the number of cluster members by removing some members in the Cluster Members and Properties page of Oracle Enterprise Manager or the Clustering page of OracleAS Web Cache Manager.

12400 HTTP request-header exceeds configured maximum individual header size (%s). Client IP: %s

error

Cause: OracleAS Web Cache received an HTTP request with an abnormally large header, indicating a possible denial-of-service attack.

Action: If the problem persists, you may need to increase the maximum individual header size in the Security page of Oracle Enterprise Manager or OracleAS Web Cache Manager.

12401 HTTP request-header exceeds configured maximum combined header size. Client IP: %s

error

Cause: OracleAS Web Cache received an HTTP request with an abnormally large header, indicating a possible denial-of-service attack.

Action: If the problem persists, you may need to increase the maximum combined header size in the Security page of Oracle Enterprise Manager OracleAS Web Cache Manager.

12402 HTTP response-header parsing error: %s.

error

Cause: An HTTP response from the origin server contained a malformed header or a individual header value that was illegal or unexpected.

Action: No action required.

12403 HTTP response-header parsing error: %s (%s).

error

Cause: An HTTP response from the origin server contained a malformed header or a individual header value that was illegal or unexpected.

Action: No action required.

12404 The number of search keys exceeds the maximum limit of search keys in the Surrogate-Key response-header.

warning

Cause: The number of search keys in the Surrogate-Key response-header exceeded the allowed configurable maximum defined in configuration file.

Action: Revise the Surrogate-Key response-header with configured maximum or fewer search keys.

12405 Illegal time format in HTTP response-header.

warning

Cause: HTTP response-header contains an illegal time format.

Action: No action required.

12409 Timeout while reading HTTP response.

error

Cause: No timely response received from the origin server, indicating that the origin server is down.

Action: Start origin server, if it is not running.

12410 Error while reading HTTP response (SSL error %d).

error

Cause: OracleAS Web Cache encountered an SSL read error.

Action: If the problem persists, contact Oracle Support.

12411 Non-parsable HTTP request-header line - questionable token syntax. Client IP: %s

warning

Cause: An HTTP request from a browser contained a line with a token value containing either an unpaired parenthesis ((or)) or an unpaired double-quote ("). For example: Cookie: name="John Hancock

Action: No action required.

12412 HTTP request-header parsing error - malformed request line. Client IP: %s

error

Cause: An HTTP request from a browser contained a malformed request line; it contained either an unknown method name or an illegal version string.

Action: No action required.

12413 Unable to get Oracle name of ESI template character set (%s).

notification

Cause: A recoverable internal error occurred while doing character set conversion.

Action: No action required.

12414 Unable to get Oracle name of ESI fragment character set (%s).

notification

Cause: A recoverable internal error occurred while doing character set conversion.

Action: No action required.

12415 Unable to do character set conversion.

warning

Cause: A system/library call has unexpectedly failed.

Action: If the problem persists, contact Oracle Support.

12416 NLS library failure while doing character set conversion.

warning

Cause: A system/library call has unexpectedly failed.

Action: If the problem persists, contact Oracle Support.

12417 HTTP request-header parsing error - no header name. Client IP: %s

error

Cause: An HTTP request from a browser contained a malformed header line; the header name delimiting colon (':') was not present.

Action: No action required.

12419 Dump of offending HTTP request-header line:

warning

Cause: Any of the causes associated with errors 12411, 12412 and 12417.

Action: No action required.

12420 The number of search keys exceeds the maximum hard limit of search keys in the Surrogate-Key response-header thus excessive search keys are discarded.

error

Cause: The number of search keys in the Surrogate-Key response-header exceeded the allowed hard maximum of 200.

Action: Revise the Surrogate-Key response-header with 200 or fewer search keys.

12421 HTTP request is invalid due to a duplicate header occurrence: "%s". Client IP: %s

error

Cause: An HTTP request from a browser contained multiple occurrences of   the same header type.

Action: No action required.

12422 Duplicate header occurrence ("%s") is in HTTP request.

trace

Cause: An HTTP request from a browser contained multiple occurrences of the same header type.

Action: No action required.

12700 Site %s did not match any defined sites, resulting in no End User Performance Monitoring for URI

trace

Cause: Normal trace behavior.

Action: No action required.

12701 Site %s matched one of the defined sites for which End User Performance Monitoring is ON.

trace

Cause: Normal trace behavior.

Action: No action required.

12702 Site %s matched a defined site with End User Performance Monitoring OFF, resulting in no End User Performance Monitoring for URI

trace

Cause: Normal trace behavior.

Action: No action required.

12703 No matching End User Performance Monitoring rule found for site %s, resulting in End User Performance Monitoring for URI.

trace

Cause: Normal trace behavior.

Action: No action required

12704 There are no End User Performance Monitoring rules defined for site %s, resulting in End User Performance Monitoring for URI.

trace

Cause: Normal trace behavior.

Action: No action required.

12705 Matching End User Performance Monitoring REGEXP rule found regexp:%s and the rule is %s

trace

Cause: Normal trace behavior. Matching End User Performance Monitoring rule found. The rule determines whether or not to monitor End User Performance for the URI.

Action: No action required.

12706 Matching End User Performance Monitoring SUBSTRING rule found. substring:%s and the rule is %s

trace

Cause: Normal trace behavior. Matching End User Performance Monitoring rule found. The rule determines whether or not to monitor End User Performance for the URI.

Action: No action required.

12712 The end-user performance monitoring file can only be set to an .html file.

error

Cause: The file specified was not an .html file.

Action: Contact Oracle Support for help in specifying the correct end-user performance monitoring file.

12713 The end-user performance monitoring file cannot be parsed. Use the following format: <SCRIPT SRC="%schronos_JavaScript_ filename"></SCRIPT> and restart OracleAS Web Cache.

error

Cause: The file could not be parsed.

Action: Use the following format for the value of the end-user performance monitoring file: <SCRIPT SRC="%schronos_JavaScript_filename"></SCRIPT>. Then, restart OracleAS Web Cache.

12714 Invalid string specified for the end-user performance monitoring file.

error

Cause: The file name specified was an invalid string.

Action: Contact Oracle Support for help in specifying the correct file.

12715 The end-user performance monitoring file contains an invalid path separator.

error

Cause: The file name specified contained an invalid path separator.

Action: Correct the file name specified.

12716 End-user performance monitoring is enabled.

notification

Cause: Normal notification behavior. End-user performance monitoring is enabled. The event is written to the log file.

Action: No action required.

12800 OracleAS Web Cache failed to read the configuration file. Dynamic changes have not been applied.

trace

Cause: Normal trace behavior.

Action: No action required.

12801 Dynamic Changes Applied Successfully

trace

Cause: Normal trace behavior.

Action: No action required.

12802 The cache server cannot dynamically mark the origin servers as running because they are not responding. Other selected dynamic changes are applied.

trace

Cause: The origin servers were not responding. The cache server disregarded this dynamic change, but applied other dynamic changes.

Action: No action required.

12803 Dynamic Changes Not Applied

trace

Cause: Normal trace behavior.

Action: No action required.

12804 Event Log rolled over.

trace

Cause: Normal trace behavior.

Action: No action required.

12805 Access log %s rolls over.

notification

Cause: Access log rolled over as a result of an on-demand rollover request.

Action: No action required.

12806 Non-existent access log %s specified in rollover request.

notification

Cause: On-demand rollover request was submitted with an invalid access log name.

Action: No action required.

12807 Event log not rolled over because the rollover file already exists.

trace

Cause: Normal trace behavior.

Action: No action required.

12808 Access Log %s not rolled over because the rollover file already exists.

notification

Cause: Access Log is rolled over as a result of on-demand rollover request.

Action: Submit another rollover request after more than one minute.

12809 Could not find the matching Origin Server entry in original configuration which the cache server is running with.

trace

Cause: Normal trace behavior.

Action: No action required.

12810 ESI fragment log for access log %s is not rolled over.

notification

Cause: Regular access log was rolled over due to an on-demand rollover request, but accompanying ESI fragment log was not rolled over for some reason.

Action: Submit another rollover request after one minute.

12811 Dynamic event log rollover fails because the event log file does not exist.

alert

Cause: Dynamic event log rollover failed because Oracle Web Cache failed to create the event log file at startup time.

Action: Review the event log file name specified in configuration.

12812 Dynamic event log rollover fails because the event log file could not be renamed.

alert

Cause: Dynamic event log rollover failed because Oracle Web Cache could not rename the file.

Action: Check the permissions of the event log file or if any other application is using the event log file.

12901 OracleAS Web Cache fails to send the message to the Oracle Web Conferencing client; redirect fails.

Error

Cause: OracleAS Web Cache could not send Oracle Web Conferencing protocol message to client.

Action: No action required.

12902 The mx list is empty in the Oracle Web Conferencing path %s.

Alert

Cause: The configured mx path set was incorrect.

Action: Configure the correct mx path.

12903 The Oracle Web Conferencing pass descriptor fails; Error: %s.

Error

Cause: OracleAS Web Cache was unable to pass descriptor to the mx server.

Action: Check the mx server configuration and log files.

12904 Oracle Web Conferencing redirect didn't occur for mx path = %s.

Error

Cause: OracleAS Web Cache was not able to pass descriptor to the mx server on any of the mx server ports.

Action: Check the mx server configuration and log files.

12905 Oracle Web Conferencing redirect is successful.

trace

Cause: Normal trace behavior.

Action: No action required

12906 The Windows socket could not be set to blocking; Error: %s.

Error

Cause: The error was caused the by the error printed in the message.

Action: Debug from the error printed in the message.

12907 The Oracle Web Conferencing URL is not valid.

Error

Cause: The URL received by webcache is invalid.

Action: Check the Oracle Web Conferencing configuration.

12908 The Oracle Web Conferencing pass descriptor is successful.

trace

Cause: Normal trace behavior.

Action: No action required.

13001 Virtual Host Map %s has a mixture of HTTP and HTTPS Protocol origin servers. All origin servers for a virtual host map must use the same protocol.

alert

Cause: The specified virtual host map consists of origin servers with HTTP and HTTPS protocols. A virtual host mapping must consist of origin servers that use either the HTTP protocol exclusively or the HTTPS protocol exclusively.

Action: From the Sites page of Oracle Enterprise Manager or Site Definitions page of OracleAS Web Cache Manager, correct the mapping. Then, restart OracleAS Web Cache.

13002 Maximum allowed incoming connections are %u

notification

Cause: Normal notification behavior indicating the maximum number of concurrent connections which will be accepted.

Action: No action required.

13003 Configuration error. Stopping admin or cache server.

alert

Cause: The admin or cache server was unable to parse or read its configuration information.

Action: Review any other error messages to determine the cause of the problem.

13007 The specified listening IP address "%s" is not valid.

alert

Cause: The specified listening IP address was not valid 32-bit dotted decimal notation, resolvable host name, ANY, or asterisk (*).

Action: 1. Check if the specified listening address is configured as a valid 32-bit numeric address written as four numbers separated by dots. Oracle Enterprise Manager permits * and OracleAS Web Cache Manager permits ANY to support all IP addresses. 2. If you need to specify host name instead of an IP address, check if the host name can be resolved to an IP address on your system. 3. If you need to rely on DNS to resolve the host name, check if DNS is correctly working.

13050 Multiple access logs use the same file name: %s

alert

Cause: Multiple access logs were configured to use the same output file name. The output file name cannot be shared by multiple access logs.

Action: Ensure that the ACCESSLOG element in the webcache.xml file has a unique FILENAME attribute.

13074 A session caching rule refers to an invalid session name: %s

alert

Cause: The session caching rule specifies a session ID which is not defined. This error usually occurs if webcache.xml has been edited manually instead of by using Oracle Enterprise Manager or OracleAS Web Cache Manager.

Action: Edit webcache.xml and set the session ID to a valid session definition.

13075 A session binding rule refers to an invalid session name: %s

alert

Cause: A session binding rule specified a session ID which was not defined. This error usually occurs if webcache.xml has been edited manually instead of by using Oracle Enterprise Manager or OracleAS Web Cache Manager.

Action: Edit webcache.xml and correct the session definition for the specified session ID.

13076 A caching rule refers to an invalid expiration rule: %s

alert

Cause: One of the caching rules contained the specified expiration rule. That expiration rule was invalid. This error usually occurs if webcache.xml has been edited manually instead of by using Oracle Enterprise Manager or OracleAS Web Cache Manager.

Action: Edit webcache.xml and find the specified expiration rule and replace it with a valid expiration rule.

13077 A caching rule refers to an invalid multi-version cookie rule: %s

alert

Cause: One of the caching rules contained the specified invalid multi-version cookie rule. This error usually occurs if webcache.xml has been edited manually instead of by using Oracle Enterprise Manager or OracleAS Web Cache Manager.

Action: Edit webcache.xml and find the multi-version cookie rule listed in the error message. Replace the rule with a valid one.

13078 A caching rule refers to an invalid session rule: %s

alert

Cause: One of the caching rules contained the specified invalid session rule. This error usually occurs if webcache.xml has been edited manually instead of by using Oracle Enterprise Manager or OracleAS Web Cache Manager.

Action: Edit webcache.xml and find the specified session rule. Replace the rule with a valid one.

13079 No matching CACHE element found in webcache.xml for current hostname (%s) and ORACLE_HOME (%s)

alert

Cause: Unable to find a CACHE element for this cache.

Action: Check that the hostname and ORACLE_HOME printed in the error message match the expected values and check the CACHE elements in webcache.xml for one which is defined for these values.

13080 No NAME attribute is specified in ROLLOVER element under ACCESSLOGSTYLES

alert

Cause: Unable to find a NAME attribute in ROLLOVER element for access logs. The ROLLOVER element for access logs must have a NAME attribute.

Action: Edit webcache.xml and find the ROLLOVER element under ACCESSLOGSTYLES which does not have NAME attribute. Add the NAME attribute to the ROLLOVER element.

13081 Any Set-Cookie is not configured with the a session binding mechanism other than Cookie-based.

alert

Cause: A session binding rule specifying a session of Any Set-Cookie was not configured with the Cookie-based session binding mechanism. Any Set-Cookie cannot use a mechanism of OC4J-based or Internal-Tracking.

Action: Change the session binding mechanism to Cookie-based.

13106 Log file rollover schedule refers to an invalid TIME value for the HOURLY frequency: %u

alert

Cause: Time value specified for the HOURLY frequency was invalid.

Action: TIME must be set to a nonzero positive integer value less than 60.

13107 Log file rollover schedule refers to an invalid TIME value for the DAILY frequency: %u

alert

Cause: Time value specified for the DAILY frequency was invalid.

Action: For DAILY rollover, the time format is "hhmm", where hh ranges from 0 to 23 and mm ranges from 0 to 59.

13108 Log file rollover schedule refers to an invalid TIME value for the WEEKLY frequency: %u

alert

Cause: Time value specified for the WEEKLY frequency was invalid.

Action: For WEEKLY rollover, the time format is "dhhmm", where "d" ranges from 0 to 6, "hh" ranges from 0 to 23 and mm ranges from 0 to 59.

13109 XML parsing error in %s. Error code %u: %s

alert

Cause: The specified XML file did not match the DTD for the file.

Action: See the related Oracle XML library error message for more details.

13110 XML parser initialization error. Error code: %u

alert

Cause: Unable to initialize the XML parser.

Action: If the problem persists, contact Oracle Support.

13111 XML parsing failed for file: %s. Error Code: %u

alert

Cause: An XML parsing error occurred when parsing the specified file.

Action: Review associated error messages for more information about the problem.

13112 XML parsing failed. Error Code: %u

error

Cause: Failed to parse an XML message.

Action: Review associated error messages for more information about the problem.

13113 Syntax error in %s. Exception: "%s" in attribute "%s" having value "%s" in element %s

error

Cause: An exception occurred while parsing the specified file.

Action: Edit the specified file to make it a valid XML object.

13119 Duplicate origin server hosts specified for host %s port %s.

error

Cause: There are duplicate entries in the HOST section of the webcache.xml file.

Action: Remove the duplicate entries from webcache.xml.

13120 CACHEWITH and CACHEWITHOUT cannot both have the value "NO" in a session caching policy for session %s.

error

Cause: The session caching policy has a value of "NO" for both CACHEWITH and CACHEWITHOUT in webcache.xml.

Action: Set at least one of them to be "YES", or remove the policy.

13121 The caching rule for %s contains more than one multi-version cookie policy based on the same cookie: %s.

error

Cause: More than one multi-version cookie policy is associated for the same cookie with one caching rule.

Action: Remove the extra policies from the caching rule.

13122 The caching rule for %s contains more than one session-caching policy based on the same session: %s.

error

Cause: More than one session caching policy is associated for the same session with one caching rule.

Action: Remove extra policies from the caching rule.

13123 Site name %s:%s%s exceeds maximum individual header limit.

error

Cause: The site name was set to more than the maximum single header limit as defined in the security section.

Action: Increase the maximum individual header size limit in the Security page of Oracle Enterprise Manager or OracleAS Web Cache Manager, or change the site name in the Sites page or Oracle Enterprise Manager or Site Definitions page of OracleAS Web Cache Manager, so that the site name and port number string together is less than the specified limit.

13124 Alias name %s:%s%s exceeds maximum individual header limit %d.

error

Cause: The alias name was set to more than maximum individual header limit as defined in the security section.

Action: Increase the maximum individual header size limit in the Security page of Oracle Enterprise Manager or OracleAS Web Cache Manager, or change the alias name in the Sites page of Oracle Enterprise Manager or Site Definitions page of OracleAS Web Cache Manager, so that the alias name and port number string together is less than the specified limit.

13125 internal.xml does not have CALYPSOINTERNALPARAMS element.

error

Cause: Unable to find CALYPSOINTERNALPARAMS element in internal.xml.

Action: Check whether internal.xml conforms to internal.dtd.

13126 No default site defined. From OracleAS Web Cache Manager, change one site from Site Definitions to be default site.

alert

Cause: All the sites defined in the configuration are set "NO" for the default site option.

Action: From OracleAS Web Cache Manager, make the changes to the site definitions so that at least one site is the default site.

13127 Oracle XML Schema processor failed to initialize%s.

error

Cause: Oracle XML Schema Processor failed to initialize.

Action: If the problem persists, contact Oracle Support.

13128 XML file%s did not pass validation against schema checking%s.

error

Cause: Error occurs in validating the XML file against its corresponding XML schema

Action: Fix the error accordingly in the XML file

13200 Access logging was skipped as the request had not been parsed properly.

error

Cause: Due to an error in request parsing, access logging could not be completed.

Action: None

13212 Access log file %s could not be opened.

alert

Cause: Access log file could not be opened.

Action: Check the status of access log file. For example, check the permission of the file.

13213 Invalid access log field name: %s

alert

Cause: An invalid access log field is specified in the configuration file webcache.xml.

Action: Set valid access log field name.

13215 I/O Error when writing access Log buffer to file. error number: %d

alert

Cause: I/O error happened when OracleAS Web Cache tried to write to the access log file.

Action: Check the status of access log file. For example, see if the disk is full.

13251 The HTML file for end-user monitoring was not found.

alert

Cause: There was a configuration error.

Action: Disable end-user monitoring or restore the HTML file that is missing.

13252 On Demand Log File Rollover for file %s was not successful. File %s could not be created.

warning

Cause: The rollover file could not be created.

Action: Check the file permissions.

13301 A failure occurred with DNS lookups. No more DNS lookups will occur.

warning

Cause: The DNS server might be unavailable.

Action: Check the DNS configuration on the local machine.

13302 Failed to open directory: %s

warning

Cause: Directory could not be opened because it does not exist or the access permissions are incorrect.

Action: Check the configuration and the permissions for accessing the directory.

13305 Failed to assign port %s: %s

alert

Cause: The selected port is being used by another application.

Action: Change the port to an unused port.

13306 Failed to configure resources for port %s: %s

alert

Cause: System listen function call failed.

Action: Restart OracleAS Web Cache.

13307 Failed to allocate resources for port, %s: %s

trace

Cause: Socket creation failed.

Action: If the problem persists, Restart OracleAS Web Cache and check open file descriptors.

13310 Problem opening file %s (Access Denied).

warning

Cause: The file could not be opened because of an access problem.

Action: Fix the file permissions so that the user running webcached can access the file.

13312 Problem opening file %s (File Already Exists).

warning

Cause: The file could not be created because it already exists.

Action: Choose a file name that does not exist.

13313 Problem opening file %s (Invalid Drive).

warning

Cause: The file could not be opened because the drive name is invalid.

Action: Correct the file specification.

13314 Problem opening file %s (Invalid Name).

warning

Cause: The file could not be opened because the file name is invalid.

Action: Correct the file specification.

13315 Problem opening file %s (Too Many Open Files).

warning

Cause: The file could not be opened because there are too many open files in the system.

Action: Close other files or programs and restart OracleAS Web Cache.

13316 Problem opening file %s (File Not Found).

warning

Cause: The file could not be opened because the file could not be found.

Action: Correct the file specification.

13317 Problem opening file %s (Sharing violation).

warning

Cause: The file could not be opened because there has been a sharing violation.

Action: Close the file through other applications that are accessing it.

13318 Problem opening file %s (Is a directory).

warning

Cause: The file could not be opened because it is directory.

Action: Correct the file specification.

13350 Error during network receive, system error %s

trace

Cause: There was system error during the networking receive.

Action: No action required.

13351 Timeout during network receive, socket number %s

error

Cause: There was timeout during the networking receive.

Action: No action required.

13365 Error during network send, system error %s

trace

Cause: There was system error during the networking send.

Action: No action required.

13366 Timeout during network send, socket number %s

trace

Cause: There was timeout during the networking send.

Action: No action required.

13380 Error during SSL handshake, system error %s

error

Cause: There was system error during the SSL handshake.

Action: No action required.

13381 Timeout during SSL handshake, socket number %s

error

Cause: There was timeout during the SSL handshake.

Action: No action required.

13382 File %s cannot be opened.

error

Cause: The specified file could not be opened.

Action: Evaluate why the specified file cannot be opened. The file may not be located in the correct directory or permissions on the file may be incorrect. Correct the problem.

13383 Unable to initialize DNS lookups

warning

Cause: Cannot use DNS for TCP/IP host resolution.

Action: No required action.

13400 parsing error of <!-- WEBCACHETAG--> and <!-- WEBCACHEEND--> at offset %d

error

Cause: An invalid token was encountered in the <!-- WEBCACHETAG--> and <!-- WEBCACHEEND--> tags.

Action: Correct the syntax of the <!-- WEBCACHETAG--> and <!-- WEBCACHEEND--> tags at the specified location in the object.

13601 Signal SIGSEGV caught

alert

Cause: Signal SIGSEGV is caught. Diagnostic information is printed in trace dump file.

Action: Contact Oracle Support.

13602 Signal SIGBUS caught

alert

Cause: Signal SIGBUS is caught. Diagnostic information is printed in trace dump file.

Action: Contact Oracle Support.

13603 Signal SIGFPE caught

alert

Cause: Signal SIGFPE is caught. Diagnostic information is printed in trace dump file.

Action: Contact Oracle Support.

13604 Signal SIGSEGV caught

alert

Cause: Signal SIGSEGV is caught. Diagnostic information is printed in trace dump file.

Action: Contact Oracle Support.

13704 Compression error in: %s

error

Cause: A gzip library call failed.

Action: If the problem persists, contact Oracle Support.

13705 OracleAS Web Cache compresses object. Original size: %d bytes; Compressed size: %d bytes.

trace

Cause: Normal trace behavior.

Action: No action required.

13708 Compression is not yielding any benefit.

trace

Cause: Compression was not able to reduce the size of the object.

Action: Disable compression for this object..

14002 Error while sending the request to origin server

warning

Cause: The request to the origin server failed.

Action: Ensure that the origin server is running.

14003 Error while receiving response from the origin server

warning

Cause: OracleAS Web Cache was unable to receive a response from the origin server.

Action: Ensure that the origin server is running.

14004 HTTP status code error

warning

Cause: The origin server returned a non-HTTP status code.

Action: Check the origin server to determine what application is sending the response.

14005 Content-Type buffer is too small

warning

Cause: The buffer pool memory was not sufficient.

Action: Adjust the maximum individual header size limit in the Security page of Oracle Enterprise Manager or OracleAS Web Cache Manager.

14006 SSL handshake with origin server fails.

warning

Cause: The SSL handshake between OracleAS Web Cache and the origin server failed.

Action: Check certificate configuration on the origin server.

14007 Connect request terminates.

warning

Cause: The connection request to the origin server was terminated.

Action: If the problem persists, contact Oracle Support.

14008 Create request fails.

warning

Cause: OracleAS Web Cache failed to create a request to the origin server.

Action: If the problem persists, contact Oracle Support.

14009 Header operation fails.

warning

Cause: OracleAS Web Cache could not write to the HTTP response header from the origin server. This occurs if the OracleAS Web Cache computer is experiencing a system memory problem.

Action: If the problem persists, contact Oracle Support.

14010 Invalid authentication credentials for proxy server

warning

Cause: The user name and password for the proxy server failed.

Action: Correct the user name and password in the Origin Servers page of Oracle Enterprise Manager or OracleAS Web Cache Manager.

14011 Proxy tunnel authentication required

warning

Cause: The user name and password for the proxy server failed.

Action: Correct the user name and password in the Origin Servers page of Oracle Enterprise Manager or OracleAS Web Cache Manager.

14012 Bad response from proxy tunnel

warning

Cause: The proxy tunnel response was incorrect.

Action: Check the proxy server configuration and setup.

14013 Send request to origin server times out.

warning

Cause: The request to the origin server exceeded the network timeout limit.

Action: If the problem persists, adjust the network timeout between OracleAS Web Cache and the origin server in the Resource Limits and Timeouts page of Oracle Enterprise Manager or Network Timeouts page of OracleAS Web Cache Manager.

14014 Receive request from origin server times out.

warning

Cause: The response from the origin server exceeded the network timeout limit.

Action: If the problem persists, adjust the network timeout between OracleAS Web Cache and the origin server in the Resource Limits page Oracle Enterprise Manager or Network Timeouts page of OracleAS Web Cache Manager.

14015 Browser terminates connection; the request to the origin server is terminated.

warning

Cause: The browser terminated its connection. As a result, OracleAS Web Cache terminated its connection to the origin server.

Action: No action required.

14016 Connection to origin server fails.

warning

Cause: The origin server is down.

Action: Check the origin server.

14017 Connection to origin server times out.

warning

Cause: The connection to the origin server exceeded the network timeout limit.

Action: If the problem persists, adjust the network timeout between OracleAS Web Cache and the origin server in the Resource Limits and Timeouts of Oracle Enterprise Manager or Network Timeouts page of OracleAS Web Cache Manager.

14018 Socket creation fails for origin server.

warning

Cause: The OracleAS Web Cache computer has run short of system resources.

Action: If the problem persists, increase the Maximum Incoming Connections setting in the Resource Limits and Timeouts page of Oracle Enterprise Manager or Resource Limits of OracleAS Web Cache Manager.

14019 Dynamic origin server DNS failure

warning

Cause: The DNS lookup for the ESI provider site failed, because either the site was down or does not exist.

Action: Ensure that the ESI provider site is operational. If it is, check the site configuration in the Sites page of Oracle Enterprise Manager or Site-to-Server Mapping page of OracleAS Web Cache Manager.

14020 Origin server rejects queue entry.

warning

Cause: The queue to the origin server was full, resulting in the new request not being accepted.

Action: If the problem persists, increase the capacity for the origin server in the Origin Servers page of Oracle Enterprise Manager or OracleAS Web Cache Manager.

14021 Capacity, as set by Web Cache, is exceeded for all origin servers that are available for processing this request.

warning

Cause: OracleAS Web Cache was unable to forward the request to the origin server, because the origin server reached its capacity and queue limit.

Action: If the problem persists, increase the capacity for the origin server in the Origin Servers page of Oracle Enterprise Manager or OracleAS Web Cache Manager.

14022 Load balancing and failover functionality fails.

warning

Cause: All of the origin servers were down. OracleAS Web Cache will continue to poll the origin server until they are back online.

Action: Check the state of the origin servers.

14023 No origin server is running.

warning

Cause: All communication to the origin servers has failed. OracleAS Web Cache will continue to poll the origin server until they are back online.

Action: Check the state of the origin servers.

14024 Dynamic origin server is not allowed for request.

warning

Cause: The request was for a virtual host site. DNS discovery of origin servers is only permitted for ESI provider sites.

Action: Check the site configuration in the Sites page of Oracle Enterprise Manager or Site Definitions and Site-to-Server Mapping page of OracleAS Web Cache Manager.

14025 Dynamic connection to origin server fails.

warning

Cause: OracleAS Web Cache received the DNS lookup result for the ESI provider site, but was unable to connect to the site.

Action: Check that the ESI provider site is operational.

14027 Connection request times out.

warning

Cause: The connection between OracleAS Web Cache and the origin server timed out.

Action: No action required.

14028 incomplete header

warning

Cause: The HTTP response from the origin server did not contain a properly formatted HTTP header.

Action: Check the application sending the response.

14029 invalid header

warning

Cause: The HTTP header response from the origin server contained an invalid field.

Action: Check the application sending the response.

14032 Buffer finished before is found.

warning

Cause: The response from the origin server was not properly formatted.

Action: If the problem persists, check the application sending the response.

14036 time overflow

warning

Cause: Invalid time string from Origin Server

Action: Check the Origin Server configuration setting

14037 illegal time format

warning

Cause: The creation time received from the origin server was for a time in the future.

Action: Check the application sending the response.

14039 HTTP response not found

warning

Cause: An HTTP response from the origin server was not found.

Action: Check the application sending the response.

14041 invalid format

warning

Cause: The response contained an invalid time format or inversion of an integer.

Action: No action required.

14044 Unexpected EOF while reading HTTP response-header.

error

Cause: Connection prematurely closed by origin server.

Action: No action required.

14045 Error Parsing Chunk Length for %s%s.

error

Cause: Incorrect chunk length

Action: No action required.

14047 OracleAS Web Cache encounters an invalid HTTP response.

error

Cause: An HTTP response from the origin server contained invalid contents. For example, the response indicated chunked contents but the response contained no chunks.

Action: Check the application sending the response.

20101 Open socket fd=%d exceeded FDArr_Size %d.

alert

Cause: Potential file descriptor leakage.

Action: Contact Oracle Support.

20150 Failure in scheduling asynchronous task: %s.

alert

Cause: Potential memory shortage. Note: This is for all asynchronous functions.

Action: Contact Oracle Support.

A

OracleAS Web Cache Directory Structure

This appendix describes the installed OracleAS Web Cache directory structure.

When you install OracleAS 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 general directory structure components of the `$ORACLE_HOME/webcache` directory on UNIX and the `ORACLE_HOME\webcache` directory on Windows. It does not provide a comprehensive list of all files.

Table A–1 OracleAS Web Cache Directory Structure

Directory/File	Contents
`/bin` directory on UNIX	Contains the following OracleAS Web Cache executables: ■ `webcached` OracleAS Web Cache executable ■ `webcachectl` executable for starting, stopping, and restarting the OracleAS Web Cache processes. ■ `webcache_setuser.sh` script for setting file permissions on UNIX **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 OracleAS Web Cache Manager, `readme.examples.html`, and `readme.toolkit.html`. The `readme.examples.html` file contains information about the source code and scripts in the `examples` directory; the `readme.toolkit.html` file contains information about the Application Program Interfaces (APIs) in the `toolkit` directory.
`/dtds` directory on UNIX `\dtds` directory on Windows	Contains the following Document Type Definition (DTD) files: ■ `wcstats.dtd` for statistic monitoring requests and responses ■ `webcache.xsd` for the `webcache.xml` file
`/examples` directory on UNIX `\examples` directory on Windows	Contains source code and scripts that enable you to better customize applications for OracleAS Web Cache features **See Also:** `readme.examples.html` in the `docs` directory for further information
`/files` directory on UNIX `\files` directory on Windows	Contains error pages as well as internal files that are used for End-User Performance Monitoring
`/lib` directory on UNIX	Contains library files
`/logs` directory on UNIX `\logs` directory on Windows	Contains event and access logs

Table A–1   (Cont.) OracleAS Web Cache Directory Structure

Directory/File	Contents
/mesg directory on UNIX \mesg directory on Windows	Contains the OracleAS Web Cache message file used for event log messages
/toolkit directory on UNIX \toolkit directory on Windows	Contains APIs that enable you to better customize applications for OracleAS Web Cache features. This directory also contains the WCSinvalidation.dtd for invalidation. **See Also:** readme.toolkit.html in the docs directory for further information
/wallets directory on UNIX \wallets directory on Windows	Contains a dummy wallet for the origin server. This wallet is intended for testing purposes. For a production environment, you must create a new wallet. **See Also:** "Task 1: Create Wallets" on page 9-1 for further information
internal.xml file	Contains internal configuration settings
webcache.pid file on UNIX	Contains the process ID of the cache server process. The admin server process, auto-restart mechanism, and the webcachectl utility read this file when working with the cache server process.
webcache.xml file	Contains configuration parameters set by OracleAS Web Cache Manager
webcache_opmn.xml file	Contains a template to add OracleAS Web Cache information to opmn.xml
webcacheadmin.pid file on UNIX	Contains the process ID of the admin server process
webcachetargets.xml file	Contains configuration parameters used by Oracle Enterprise Manager

B

OracleAS Web Cache as a Standalone Product

This appendix explains how to install a standalone OracleAS Web Cache. It also describes the differences in how you administer a standalone OracleAS Web Cache from an Oracle Application Server installation.

This appendix contains these topics:

- Installing Standalone OracleAS Web Cache

- Differences When OracleAS Web Cache Is Installed Standalone

- OracleAS Web Cache Processes

- webcachectl Utility Overview

Installing Standalone OracleAS Web Cache

You can install OracleAS Web Cache on a dedicated computer without Oracle Application Server by performing a standalone installation. Standalone OracleAS Web Cache is distributed on the OracleAS Companion CD, which is included in the Oracle Application Server CD Pack.

1. Insert the OracleAS Companion CD, and perform the following to launch Oracle Universal Installer to install OracleAS Web Cache.

   - On UNIX:

     ```
     prompt > cd
     prompt > mount_point/1012disk1/runInstaller
     ```

   - On Windows:

     If your computer supports the auto-run feature, then the installer launches automatically.

     If your computer does not support auto-run, then double-click the setup.exe file to launch the installer.

   Oracle Universal Installer launches.

2. When the Oracle Universal Installer appears, review the Welcome screen, and click **Next**.

3. If this is the first time you are installing any Oracle products on your computer, the Specify Inventory Directory and Credentials screen appears.

   Enter the following information:

- **Enter the full path of the inventory directory**: Enter a full path to the inventory directory. Enter a directory that is different from the Oracle home directory for the product files.

  Example: `/opt/oracle/oraInventory`

- **Specify Operating System group name**: Select the operating system group that will have write permission for the inventory directory.

  Example: `oinstall`

Click **Next**. A window appears and asks you to run `orainstRoot.sh`. Run the script in a different shell as the root user. The script is located in the `oraInventory` directory. Click **Continue**. The Specify File Locations screen appears.

4. Enter the following destination information:

- **Name**: Enter a name to identity this Oracle home. The name can consist of alphanumeric and the underscore (_) characters only, and cannot be longer than 128 characters.

  Example: `OH_STANDWC`

- **Path**: Enter the full path to the destination directory. This path is the Oracle home. If the directory does not exist, the installer creates it. If you want to create the directory beforehand, create it as the `oracle` user; do not create it as the root user.

  Example: `/opt/oracle/STANDWC`

  Click **Next**. The Select a Product to Install screen displays.

5. Select **Web Server Services**, and click **Next**. The Select Installation Type screen displays.

6. Select the **OracleAS Web Cache 10.1.2.0.2** option, and click **Next**. The Specify OracleAS Web Cache Administrator Password appears.

7. In the **OracleAS Web Cache administrator Password** and **Confirm Password** fields, enter the password for the administrative accounts, and click **Next**. The Summary screen displays.

   This password applies to the Oracle Application Server administrative usernames, `ias_admin` and the OracleAS Web Cache administrative user names, `administrator` and `invalidator`. In a cache cluster, you must configure the same password for all cluster members.

   > **See Also:**
   >
   > - Online help for this page for further information about setting the password
   >
   > - "Classes of Users and Their Privileges" on page 4-6 for further information about the administrative accounts
   >
   > - "Task 2: Modify Security Settings" on page 8-7 for instructions on changing the `administrator` and `invalidator` passwords

8. Verify your selection, and click **Install**.

   The Install Progress screen displays the progress of the installation.

   After installation, the Configuration Assistants screen displays.

9. Monitor the progress of the OracleAS Web Cache Configuration Assistant. This assistant configures OracleAS Web Cache. You will be prompted to run `root.sh`. Run the script in a different shell as the root user. Click **OK**.

   The End of Installation screen displays once installation is complete.

10. Record the URL for accessing OracleAS Web Cache Manager.

11. Click **Exit** to quit the installer.

> **See Also:** *Oracle Application Server Installation Guide* for more information about the Oracle Universal Installer

Post-Installation Tasks

During the installation process, the OracleAS Web Cache Configuration Assistant establishes a default configuration.

> **See Also:** "Using the Default Configuration" on page 8-1

In order to configure OracleAS Web Cache, the Assistant assumes a dummy origin server resides on the local computer. The Oracle Universal Installer configures ports as follows:

- On UNIX systems, the Assistant assigns port 7777 to the OracleAS Web Cache listen port and port 80 to the origin server.

- On Windows systems, for a first-time install, the Assistant assigns port 80 to both the OracleAS Web Cache listening port and the origin server. Upon subsequent installs, the Assistant assigns port 7777 to the OracleAS Web Cache listening port and port 80 to the origin server.

If these ports are in use, then the Oracle Universal Installer attempts to assign another port number from a range of 7777 to 7877.

After you install OracleAS Web Cache, reconfigure the settings for the origin server to reflect the actual origin server in your environment.

> **See Also:** "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25

Differences When OracleAS Web Cache Is Installed Standalone

If you are running OracleAS Web Cache in a standalone environment, note the following differences in how you administer OracleAS Web Cache:

- OracleAS Web Cache has three processes, rather than two processes. On Windows, two of these processes are Windows services. See "OracleAS Web Cache Processes" on page B-4 for more information.

- You use the OracleAS Web Cache Manager rather than Application Server Control Console to configure OracleAS Web Cache. See "Overview of Tools for Standalone Configurations" on page 6-11.

- You use the `webcachectl` utility rather than the Oracle Process Manager and Notification (OPMN) Server, to start, stop, and restart the OracleAS Web Cache processes. (In most cases, you can also use the OracleAS Web Cache Manager to start, stop, and restart the processes.) See "webcachectl Utility Overview" on page B-4 for more information.

OracleAS Web Cache Processes

In a standalone environment, OracleAS Web Cache has three processes:

- The `admin` server process manages the OracleAS Web Cache Manager interface.

  In standalone environments on Windows, the `admin` server process is represented by the Oracle*HOME_NAME*WebCacheAdmin service.

- The `cache` server process manages the cache.

  In standalone environments 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. (In an environment where you have installed OracleAS Web Cache as part of an Oracle Application Server installation, the auto-restart mechanism is controlled by OPMN and is not a process.)

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

  > **See Also:** "Task 3: Configure Auto-Restart Settings" on page 8-10 for instructions on enabling auto-restart

webcachectl Utility Overview

In most cases, you can use the OracleAS Web Cache Manager to start, stop, and restart the processes. However, in previous releases, OracleAS Web Cache also provided the `webcachectl` utility to start, stop, and restart the `admin` server process, the `cache` server process, and the `auto-restart` process. Beginning with OracleAS Web Cache 10*g* (9.0.4), when OracleAS Web Cache is installed as part of an Oracle Application Server installation, you must use OPMN rather than the `webcachectl` utility.

If you are running OracleAS Web Cache in a standalone environment (that is, you installed OracleAS Web Cache from a kit that included only this product; you did not install OracleAS Web Cache as part of an Oracle Application Server installation), you use the `webcachectl` utility to administer the OracleAS Web Cache processes: the `admin` server, `cache` server, and `auto-restart` processes.

> **Note:** If you invoke the `webcachectl` utility in an environment where you have installed OracleAS Web Cache as part of an Oracle Application Server installation, OracleAS Web Cache returns an error.
>
> See:
>
> - "Managing Processes with Oracle Process Manager and Notification (OPMN)" on page 6-8
> - "Starting and Stopping OracleAS Web Cache In an Oracle Application Server Configuration" on page 7-1 to start or stop OracleAS Web Cache using OPMN

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

The syntax for this utility is as follows:

```
webcachectl command [parameter]
```

In standalone environments on Windows, you can also start or stop OracleAS 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 click **Start** or **Stop.**

3. Select the Oracle*HOME_NAME*WebCache service to start the `cache` server process, and then click **Start** or **Stop.** This service also starts and stops the auto-restart process.

4. In the Services window, click **Close.**

webcachectl Utility Commands

The commands for the `webcachectl` utility are described in the next sections. 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 OracleAS Web Cache Manager. After OracleAS Web Cache Manager configuration is complete, the `admin` server process is no longer needed unless you want to monitor the cache using the OracleAS Web Cache Manager.

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

  The `cache` server process is the only process required to run the cache.

reset

Use the `reset` command to restore the configuration to the last version saved with **Apply Changes** button in the OracleAS Web Cache Manager. This command also stops any running processes.

The following message displays:

```
Previous configuration restored.
You must restart OracleAS 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.
```

restart

Use the `restart` command to stop and then restart the `admin` server, `cache` server, and, if enabled, the `auto-restart` processes. The following message displays:

```
Web Cache admin server stopping.
Web Cache auto-restart monitor stopping.
Web Cache cache server stopping.

Oracle Application Server Web Cache 10g (10.1.2)
Copyright (c) 1999, 2005, Oracle. All rights reserved.
Admin Server now running as process 17722

Admin Server is attempting to start the Cache Server
Cache Server now running as process 17724
```

restartadm

Use the `restartadm` command to stop and then restart the `admin` server process. The following message displays:

```
Web Cache admin server is already down.

Oracle Application Server Web Cache 10g (10.1.2)
Copyright (c) 1999, 2005, Oracle. All rights reserved.
Admin Server now running as process 17729

Admin Server running in admin-only mode. Cache Server NOT started
```

restartcache

Use the `restartcache` command to stop and then restart the `cache` server process and, if enabled, the `auto-restart` process. The following message displays:

```
Web Cache admin server stopping.
Web Cache auto-restart monitor is already down.
Web Cache cache server stopping.

Oracle Application Server Web Cache 10g (10.1.2)
Copyright (c) 1999, 2005, Oracle. All rights reserved.

Cache Server now running as process 21910
```

start

Use the `start` command to start the `admin` server, `cache` server process, and, if enabled, the `auto-restart` process. The following message displays:

```
Oracle Application Server Web Cache 10g (10.1.2)
Copyright (c) 1999, 2005, Oracle. All rights reserved.
Admin Server now running as process 8911

Admin Server is attempting to start the Cache Server
Cache Server now running as process 8913
```

startadm

Use the `startadm` command to start the `admin` server process. The following message displays:

```
Oracle Application Server Web Cache 10g (10.1.2)
Copyright (c) 1999, 2005, Oracle. All rights reserved.
Admin Server now running as process 8971

Admin Server running in admin-only mode. Cache Server NOT started
```

startcache

Use the `startcache` command to start the `cache` server process and, if enabled, the `auto-restart` process. The following message displays:

```
Oracle Application Server Web Cache 10g (10.1.2)
Copyright (c) 1999, 2005, Oracle. All rights reserved.
Cache Server now running as process 17752
```

status

Use the `status` command to find out if the `admin` server process, `cache` server process, and, if enabled, the `auto-restart` process, are running or not running.

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:

```
Web Cache admin server is running as process 16274.
Web Cache auto-restart is running as process 16275.
Web Cache cache server is running as process 16273.
```

stop

Use the `stop` command to stop the `admin` server process, `cache` server process, and, if enabled, the `auto-restart` process. With the `stop` command, the `cache` server process does not accept any new connections, but it completes the requests of all existing connections before it shuts down. The following message displays:

```
Web Cache admin server stopping.
Web Cache auto-restart monitor is already down.
Web Cache cache server stopping.
```

stopabort

Use the `stopabort` command to stop the `admin` server process, `cache` server process, and, if enabled, the `auto-restart` process. With the `stopabort` command, the `cache` server process does not accept any new connections, drops all existing connections, and shuts down.

The following message displays:

```
Web Cache admin server stopping.
Web Cache auto-restart monitor stopping.
Web Cache cache server stopping.
```

stopadm

Use the `stopadm` command to stop the `admin` server process. The following message displays:

```
Web Cache admin server stopping.
```

stopcache

Use the `stopcache` command to stop the `cache` server process and, if enabled, the `auto-restart` process. With the `stopcache` command, the `cache` server process does not accept any new connections, but it satisfies the requests of all existing connections before it shuts down. The following message displays:

```
Web Cache auto-restart monitor stopping.
Web Cache cache server stopping.
```

webcachectl Parameter

The `webcachectl` utility supports the following parameter intended for Oracle Support Services:

`coreok`: Enables OracleAS Web Cache to produce a core dump

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
- Statistics DTD

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/toolkit` directory on UNIX and the *ORACLE_ HOME*`\webcache\toolkit` directory on Windows.

This section contains the following topics:

- Invalidation Request and Response DTD
- Invalidation Preview Request and Response DTD

Invalidation Request and Response DTD

Example C–1 shows the portion of the DTD for invalidation requests.

Example 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          #OPTIONAL
             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 shows elements and attributes of the invalidation request DTD.

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		Optional element in the INVALIDATION element

Table C–1   (Cont.) Invalidation Request DTD Elements and Attributes

Element	Attribute	Description
SYSTEMINFO		Required element in the SYSTEM element
		The possible NAME/VALUE pairs are 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.
		■ NAME="WCS_DISCONNECTED_MODE_OK" VALUE="TRUE|FALSE"
		This pair specifies how soon invalidation takes place. If WCS_DISCONNECTED_MODE_OK is TRUE, invalidation is not immediately performed. The invalidation response is sent as soon as the invalidation request is received. Set this element to TRUE, if you do not want to wait for the invalidation result. If WCS_DISCONNECTED_MODE_OK is FALSE, invalidation is completed immediately and the invalidation result is sent.
OBJECT		Required element in the invalidation request. You can specify more than one OBJECT element in the request
	BASICSELECTOR	Invalidation based on the URL
	ADVANCEDSELECTOR	Invalidation based on advanced invalidation selectors
	ACTION	Action to perform on objects
BASICSELECTOR		Optional element
	URI	URL of the objects to be invalidated
ADVANCEDSELECTOR		Optional element
	URIPREFIX	Path prefix of the objects to be invalidated
		The path prefix 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 objects 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 requests.
	METHOD	**HTTP request method** of the objects to be invalidated. The default is GET.
	BODYEXP	HTTP POST body message of the objects to be invalidated
COOKIE		Optional element
	NAME	Cookie name used by the objects contained within the URL
	VALUE	Value of the cookie
		If no value is present, then only objects with the named cookie but without value are invalidated.
HEADER		Optional element

Table C–1   (Cont.) Invalidation Request DTD Elements and Attributes

Element	Attribute	Description
	NAME	**HTTP request header** used by the objects contained within the URL
	VALUE	Value of the request header
OTHER		Optional element
	NAME	NAME is one of the following: ■ URI to specify a match of the URL ■ BODY to specify a match of the HTTP POST body ■ QUERYSTRING_PARAMETER to specify a match of an embedded URL parameter ■ SEARCHKEY to specify a match of a search key in the Surrogate-Key response header
	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, and SEARCHKEY
ACTION		Optional element
	REMOVALTTL	Maximum time that objects can reside in the cache before they are invalidated. The default is 0 seconds.
INFO		Optional element
	VALUE	Comment to be included in the invalidation result

Example C–2 shows the portion of the DTD for invalidation responses.

Example 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 shows elements and attributes of the invalidation response DTD.

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		Optional element
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		Required element in the `SYSTEM` element
	BASICSELECTOR	Invalidation based on the URL
	ADVANCEDSELECTOR	Invalidation based on advanced invalidation selectors
	RESULT	Action performed on objects
	INFO	Comment specified in the `INFO` element of the invalidation request
BASICSELECTOR		**See:** "BASICSELECTOR" on page C-3
ADVANCEDSELECTOR		**See:** "ADVANCEDSELECTOR" on page C-3
RESULT		
	ID	Sequence number of all the URLs sent in the invalidation response. If there are multiple URLs specified in the invalidation message, 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 objects that are not cacheable ■ `URI NOT FOUND` for objects not found
	NUMINV	Number of objects invalidated

Invalidation Preview Request and Response DTD

Example C–3 shows the portion of the DTD for invalidation preview requests.

Example 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 shows elements and attributes of the invalidation preview request DTD.

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 object to be listed
	MAXNUM	Number of objects to be listed
SYSTEM		Optional element
SYSTEMINFO		The possible `NAME`/`VALUE` pair are 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. ■ `NAME="WCS_DISCONNECTED_MODE_OK" VALUE="TRUE|FALSE"` This pair specifies how soon invalidation takes place. If `WCS_DISCONNECTED_MODE_OK` is `TRUE`, invalidation is not immediately performed. The invalidation response is sent as soon as the invalidation request is received. Set this element to `TRUE`, if you do not want to wait for the invalidation result. If `WCS_DISCONNECTED_MODE_OK` is `FALSE`, invalidation is completed immediately and the invalidation result is sent.
BASICSELECTOR		**See:** "BASICSELECTOR" on page C-3
ADVANCEDSELECTOR		**See:** "ADVANCEDSELECTOR" on page C-3

Example C–4 shows the portion of the DTD for invalidation preview responses.

Example 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 on page C-7 shows elements and attributes of the invalidation preview response DTD.

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 objects that are not cacheable ■ `URI NOT FOUND` for objects not found
	STARTNUM	Number representing the first object 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:**"BASICSELECTOR" on page C-3
ADVANCEDSELECTOR		**See:** "ADVANCEDSELECTOR" on page C-3

Statistics DTD

OracleAS 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 contains the following topics:

■ Statistics Request and Response DTD

■ Groups of Statistics

■ Query Methods

■ Statistics Examples

■ Complete Statistics Template

Statistics Request and Response DTD

The DTD for both statistics 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.

Example C–5 shows the contents of the statistics DTD file, `wcstats.dtd`.

Example 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 for each element and 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, OracleAS 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. Table C–6 lists the names and IDs of each top-level group.
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 the group URL_STATS and specify how many URLs are to be returned in the result. You pass the string "OBJECT_COUNT" as the NAME of the parameter 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 OracleAS 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 OracleAS Web Cache will add new groups only after existing groups, without changing the existing order. If, in future versions, OracleAS Web Cache deletes any existing groups, that group ID will not be reused for other purposes.

Conceptually, these groups are divided into six main categories. Table C–6 shows the group names, their corresponding group IDs and the category of the group.

Table C–6   Groups of Statistics

Group Name	Group ID	Category of Group
APP_SRVR_REQUEST_BACKLOG	120	Runtime Statistics (general)
APP_SRVR_STATS	123	Origin Server Statistics
BYTES_SAVED_WITH_COMPRESSION	105	Runtime Statistics (timed)
BYTES_SERVED	104	Runtime Statistics (timed)
CACHE_INFO	133	Cache Information
CACHE_REASONS	135	Cache Reasons Group
CACHE_REDIRECT_DOC_COUNT	122	Runtime Statistics (general)
CACHEABILITY_RULES	125	Cache Information
CACHED_DOC_COUNT	108	Cache Information
CACHED_DOC_SIZE	109	Cache Information
CLUSTERS	130	Cache Information
COMPRESSED_HITS	115	Runtime Statistics (timed)
COMPRESSED_MISSES	116	Runtime Statistics (timed)
ERRORS	127	Runtime Statistics (timed)
HITS	128	Runtime Statistics (timed)
HTTP_CLIENT_REQUESTS	132	Runtime Statistics (timed)
HTTP_REQUESTS	126	Runtime Statistics (timed)
INVALIDATED_OBJECTS	107	Runtime Statistics (timed)
INVALIDATION_REQUESTS	106	Runtime Statistics (timed)
MISSES	129	Runtime Statistics (timed))
OPEN_CONNECTIONS	103	Runtime Statistics (general)
PID	102	Cache Information
REFRESHES	114	Runtime Statistics (timed)
SESSION_COUNT	121	Runtime Statistics (general)
SITE_LIST	134	Site Information
TIME	101	Cache Information
URL_STATS	124	URL Statistics

The following sections describe each category:

- Cache Information Groups

- Runtime Statistics Groups

- Site Information Groups

- Origin Server Statistics Group

- URL Statistics Group

- Cache Reasons 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 Groups

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 objects in the cache
	CURRENT	The total number of objects currently stored in the cache. CURRENT returns the aggregate of the owned and on-demand objects.
OWNED		A subgroup of CACHED_DOC_COUNT
	CURRENT	The number of owned objects currently stored in the cache
DEMAND		A subgroup of CACHED_DOC_COUNT
	CURRENT	The number of on-demand objects currently stored in the cache
CACHED_DOC_SIZE		A group that returns information about the size of objects in the cache
	CURRENT	The size, in bytes, of the contents of the objects currently stored in the cache. CURRENT returns the aggregate of the owned and on-demand objects.
OWNED		A subgroup of CACHED_DOC_SIZE
	CURRENT	The size, in bytes, of the content of the owned objects currently stored in the cache
DEMAND		A subgroup of CACHED_DOC_SIZE

Table C–7   (Cont.) Cache Information Groups

GROUP Name	ENTRY Name	Description
	CURRENT	The size, in bytes, of the content of the on-demand objects currently stored in the cache
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-five 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 OracleAS Web Cache for cache storage and operation
	VERSION	The version of OracleAS Web Cache
CACHEABILITY_ RULES		A group that returns information about the configured caching rules
RULE		A subgroup of CACHEABILITY_RULES
	INDEX	An index generated by OracleAS Web Cache that represents the caching rule
	EXPRESSION_TYPE	The type of expression specified for the caching rule. Valid values are: PATH_PREFIX: A prefix for the path FILE_EXT: A file extension REGEXP: A regular expression
	REGULAR_ EXPRESSION	The expression specified for the caching rule. This entry may be deprecated in a future release. It is replaced by the entry EXPRESSION.
	EXPRESSION	The expression specified for the caching rule. This entry means the same as the entry REGULAR_EXPRESSION, which may be deprecated in a future release.
	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 OracleAS 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 OracleAS Web Cache that indicates the version of the configuration file

Runtime Statistics Groups

OracleAS 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 Groups

GROUP Name	PARAM Name	ENTRY Name	Description
OPEN_CONNECTIONS	none		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	none		A group that returns information about origin server request backlogs
		CURRENT	The current number of requests that the origin server is processing for OracleAS Web Cache
		MAX_SINCE_START	The maximum number of requests that the origin server has processed for OracleAS Web Cache since the cache was last started
SESSION_COUNT			A group that returns information about the open sessions
	SITE_NAME[1]		Host name and port number of the site (*host_name:port*)
	SITE_ID[1]		An index generated by OracleAS Web Cache as an identifier for the site
		CURRENT	The number of current open sessions
		MAX_SINCE_START	The maximum number of sessions that OracleAS Web Cache had open at any one time since it was last started
CACHE_REDIRECT_DOC_COUNT			A group that returns information about object redirection
	SITE_NAME[1]		Host name and port number of the site (*host_name:port*)
	SITE_ID[1]		An index generated by OracleAS Web Cache as an identifier for the site
		CURRENT	The number of objects currently redirected by OracleAS Web Cache
		MAX_SINCE_START	The maximum number of objects redirected by OracleAS Web Cache since the cache was last started

[1]   If you specify a PARAM of SITE_NAME or SITE_ID, OracleAS Web Cache returns the statistics for that site. If you do not specify a PARAM of SITE_NAME or SITE_ID, OracleAS Web Cache returns statistics for all sites. A SITE_ID of 0 indicates unnamed sites.

Table C–9 lists the timed runtime statistics groups, subgroups, and parameters, 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 on page C-13:

- `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 Groups

Group Name	PARAM Name	Description
BYTES_SAVED_WITH_ COMPRESSION		The ENTRY elements for this group return statistics about the additional bytes that would be sent to browsers if in-cache compression is turned off.
	SITE_NAME[1]	Host name and port number of the site (*host_name*:*port*)
	SITE_ID[1]	An index generated by OracleAS Web Cache as an identifier for the site
BYTES_SERVED		The ENTRY elements for this group return statistics about the bytes served by the cache.
	SITE_NAME[1]	Host name and port number of the site (*host_name*:*port*)
	SITE_ID[1]	An index generated by OracleAS Web Cache as an identifier for the site
HTTP_CLIENT_REQUESTS		The ENTRY elements for this group return statistics about the browser and peer cache requests served by the cache. This does not include ESI requests.
	SITE_NAME[1]	Host name and port number of the site (*host_name*:*port*)
	SITE_ID[1]	An index generated by OracleAS Web Cache as an identifier for the site
HTTP_REQUESTS		The ENTRY elements for this group return statistics about the browser, peer cache, and ESI requests served by the cache.
	SITE_NAME[1]	Host name and port number of the site (*host_name*:*port*)
	SITE_ID[1]	An index generated by OracleAS Web Cache as an identifier for the site
INVALIDATED_OBJECTS		The ENTRY elements for this group return statistics about the invalidated objects.
	SITE_NAME[1]	Host name and port number of the site (*host_name*:*port*)
	SITE_ID[1]	An index generated by OracleAS Web Cache as an identifier for the site
INVALIDATION_REQUESTS		The ENTRY elements for this group return statistics about the invalidation requests submitted to the cache.
	SITE_NAME[1]	Host name and port number of the site (*host_name*:*port*)
	SITE_ID[1]	An index generated by OracleAS Web Cache as an identifier for the site

Table C–9   (Cont.) Timed Runtime Statistics Groups

Group Name	PARAM Name	Description
HITS		The ENTRY elements for this group return statistics about the client requests resolved by objects stored in the cache.
		Every ENTRY element in this group returns the aggregate value of the corresponding ENTRY elements in the subgroups.
	SITE_NAME[1]	Host name and port number of the site (*host_name:port*)
	SITE_ID[1]	An index generated by OracleAS Web Cache as an identifier for the site
FRESH_HITS		A subgroup of the group HITS
		The ENTRY elements for this group return statistics about the client requests resolved by valid, fresh objects stored 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 client requests resolved by owned objects in the cache.
FROM_OWNED_TO_PEER		A subgroup of the group FRESH_HITS
		The ENTRY elements for this group return statistics about client requests resolved by retrieving owned objects from a peer cache.
FROM_DEMAND_TO_CLIENT		A subgroup of the group FRESH_HITS
		The ENTRY elements for this group return statistics about client requests resolved by on-demand content.
STALE_HITS		A subgroup of the group HITS
		The ENTRY elements for this group return statistics about client 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 client 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 client requests resolved by expired or invalidated owned objects from a peer cache.
FROM_DEMAND_TO_CLIENT		A subgroup of the group STALE_HITS
		The ENTRY elements for this group return statistics about client requests resolved by expired or invalidated on-demand content.
MISSES		The ENTRY elements for this group return statistics about cacheable and non-cacheable misses. Misses are client requests for objects 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.
	SITE_NAME[1]	Host name and port number of the site (*host_name:port*)

Table C–9   (Cont.) Timed Runtime Statistics Groups

Group Name	PARAM Name	Description
	SITE_ID[1]	An index generated by OracleAS Web Cache as an identifier for the site
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 client requests for cacheable owned objects that were not served by the cache.
FROM_OWNED_TO_PEER		A subgroup of the group CACHEABLE_MISSES
		The ENTRY elements for this group return statistics about client requests for objects 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 non-cacheable misses. Non-cacheable misses are requests for objects that should not be cached.
		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 non-cacheable misses for owned objects.
FROM_OWNED_TO_PEER		A subgroup of the group NONCACHEABLE_MISSES
		The ENTRY elements for this group return statistics about non-cacheable misses for owned objects not served to a peer cache.
FROM_DEMAND_TO_CLIENT		A subgroup of the group NONCACHEABLE_MISSES
		The ENTRY elements for this group return statistics about non-cacheable misses for on-demand objects that were not served by the cache.
OWNER_UNKNOWN		A subgroup of the group NONCACHEABLE_MISSES
		The ENTRY elements for this group return statistics about non-cacheable misses for which the owner is unknown.
REFRESHES		The ENTRY elements for this group return statistics about the objects that the cache has refreshed from the application Web servers.
	SITE_NAME[1]	Host name and port number of the site (*host_name:port*)
	SITE_ID[1]	An index generated by OracleAS Web Cache as an identifier for the site
COMPRESSED_HITS		The ENTRY elements for this group return statistics about the total requests served from the cache in compressed form.
	SITE_NAME[1]	Host name and port number of the site (*host_name:port*)

Table C–9 (Cont.) Timed Runtime Statistics Groups

Group Name	PARAM Name	Description
	SITE_ID[1]	An index generated by OracleAS Web Cache as an identifier for the site
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.
	SITE_NAME[1]	Host name and port number of the site (*host_name:port*)
	SITE_ID[1]	An index generated by OracleAS Web Cache as an identifier for the site
ERRORS		The ENTRY elements for this group return statistics about the error pages that the cache has served.
		Every ENTRY element in this group returns the aggregate value of the corresponding ENTRY elements in the subgroups.
	SITE_NAME[1]	Host name and port number of the site (*host_name:port*)
	SITE_ID[1]	An index generated by OracleAS Web Cache as an identifier for the site
NETWORK_ERRORS		A subgroup of the group ERRORS
		The ENTRY elements for this group return statistics about the error pages that the cache has served to clients due to a network error.
SITE_BUSY_ERRORS		A subgroup of the group ERRORS
		The ENTRY elements for this group return statistics about the error pages that the cache has served to clients 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 the sum of ESI_UNCAUGHT_EXCEPTIONS and ESI_DEFAULT_FRAGMENT_SERVED.
ESI_UNCAUGHT_EXCEPTIONS		A subgroup of the group ERRORS
		The ENTRY elements for this group return statistics about the number of errors that the cache has returned to a client when an uncaught exception occurred in the cache during ESI parsing or processing.
ESI_DEFAULT_FRAGMENT_SERVED		A subgroup of the group ERRORS
		The ENTRY elements for this group return statistics about the error pages that the cache has served when an uncaught exception occurs in the application during ESI parsing or processing.

[1] If you specify a PARAM of SITE_NAME or SITE_ID, OracleAS Web Cache returns the statistics for that site. If you do not specify a PARAM of SITE_NAME or SITE_ID, OracleAS Web Cache returns statistics for all sites. A SITE_ID of 0 indicates unnamed sites.

Site Information Groups

The site information groups provide multiple-site support. You can request statistics for all sites or for a specific site. OracleAS 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 groups, the valid values that can be passed to the NAME attribute of the ENTRY element for the groups, and a description of the attribute.

Table C–10   Site Information Groups

GROUP Name	ENTRY Name	Description
SITE_LIST		A group that returns information about the sites hosted by OracleAS Web Cache
SITE		A subgroup of SITE_LIST. It returns information about a Web site hosted by OracleAS Web Cache.
	NAME	Host name and port number of the site (*host_name:port*)
	ID	An index generated by OracleAS Web Cache as an identifier of the site

You can obtain the following general runtime statistics for all sites or for a specific site:

- SESSION_COUNT

- CACHE_REDIRECT_DOC_COUNT

See Table C–8 for information about the statistics and the parameters used to obtain the statistics for sites.

You can obtain the following timed runtime statistics for all sites or for a specific site:

- BYTES_SAVED_WITH_COMPRESSION

- BYTES_SERVED

- COMPRESSED_HITS

- COMPRESSED_MISSES

- ERRORS

- HITS

- HTTP_CLIENT_REQUESTS

- HTTP_REQUESTS

- INVALIDATED_OBJECTS

- INVALIDATION_REQUESTS

- MISSES

- REFRESHES

See Table C–9 for information about the statistics and the parameters used to obtain the statistics for sites.

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 group 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 OracleAS Web Cache requests
	IS_PROXY	Whether or not the origin server is a proxy server. Possible values are YES and NO.
	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
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 OracleAS Web Cache
	MAX_PER_SEC_SINCE_START	The maximum number of seconds used to process requests for OracleAS Web Cache since the origin server was last restarted
	AVG_PER_SEC_SINCE_START	The average number of seconds used to process requests for OracleAS Web Cache since the origin server was last restarted
	TOTAL_SINCE_START	The total number of seconds used to process requests for OracleAS Web Cache since the origin server was last restarted
	TOTAL_SINCE_RESET	The total number of seconds used to process requests for OracleAS Web Cache since the statistics were reset
	AVG_PER_SEC_SINCE_RESET	The average number of seconds used to process requests for OracleAS Web Cache since the statistics were reset
ACTIVE_SESSIONS		A subgroup of the group SERVER
	CURRENT	The number of current active connections from OracleAS 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 active connections from OracleAS 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 OracleAS Web Cache that the origin server has open
	MAX_SINCE_START	The maximum number of connections from OracleAS 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 requested objects. You specify the number of URLs to be returned. This information is returned in the Popular Requests page of Application Server Control Console (**Web Cache Home** page > **Performance** tab > **All Sites** section > **Popular Requests** link) or OracleAS Web Cache Manager (**Monitoring** > **Popular Requests**).

The URL statistics group can contain multiple URL subgroups.

Table C–12 lists the URL statistics group and subgroups, the valid values that can be passed to the NAME attribute of the ENTRY element or PARAM for the group, and a description of the attribute.

Table C–12   URL Statistics Group

GROUP Name	PARAM Name	ENTRY Name	Description
URL_STATS			A group that returns information, such as the URL, about the most requested objects.
	OBJECT_COUNT		The number of URLs to be returned. You must supply a number to the VALUE attribute.
	TYPE		The type of requests. Possible values are: CACHED: Returns the URLs of the most popular cached objects NOT_CACHED: Returns the URLs of the most popular objects that were not cached ALL: Returns the most popular objects, cached and not cached
URL			A subgroup of the group URL_STATS
		NAME	The URL of the request
		SCORE	An internally generated, relative value that indicates the popularity of a URL
		CACHABILITYRULE	The index to the caching rule that triggered the object to be cached
		SIZE	The size, in bytes, of the object represented by the URL. If the size is unknown (because the object is currently in the process of being returned), the value -1 is returned.

Table C–12   (Cont.) URL Statistics Group

GROUP Name	PARAM Name	ENTRY Name	Description
		TYPE	Whether the object is cached or not cached. Possible values are: CACHED NOT_CACHED CACHED_VALIDATION_NEEDED
		REASON	The reason that the object is cached or not cached. Possible reasons are listed in Table C–13.
		COMPRESSED	Whether or not the object was compressed when it was served. Possible values are YES and NO.

Cache Reasons Group

The cache reasons group provides information about why an object is cached or not cached.

Table C–13 lists the cache reasons 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–13   Cache Reasons Group

GROUP Name	ENTRY Name	Description
CACHE_REASONS		A group that returns information about why an object is cached or not cached
	REASON	The reason that the object is cached or not cached. Possible values are: ■ BY_SURROGATE_CONTROL_HEADER: Cached or not cached because of information in the Surrogate-Control response header. ■ BY_REFERENCE_TTL: Cached because of the nonzero value of the reference TTL (time-to-live parameters) specified in the ESI tag. ■ BY_VALIDATION: Cached because of the ETag response header. ■ BY_X_ORACLE_HEADERS: Cached or not cached because of information in the X-Oracle-Cache response header. ■ BY_CACHING_RULE: Cached or not cached because of the caching rule. ■ BY_HTTP_HEADERS: Cached or not cached because of information in the HTTP header. ■ BY_PRESEEDING: Cached because the objects were preseeded for use by end-user performance monitoring. See "Configuring End-User Performance Monitoring" on page 14-3 for information about end-user performance monitoring. ■ NC_NO_DIRECTIVE: Not cached because no directive or rule has stated that the object should be cached. ■ NC_SIZE_TOO_LARGE: Not cached because the object is larger than the size specified as the maximum cached object size. See "Maximum Size of Single Cached Object" on page 8-18 for more information about the maximum cached object size. ■ NC_COOKIE_MISMATCH: Not cached because the response contains a cookie that is not present in the request or that has a different value than the same cookie in the request.

Query Methods

To retrieve OracleAS Web Cache statistics, you send a `POST` message to the OracleAS Web Cache statistics port. By default, the statistics port is 9402.

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.

Statistics Examples

The following examples illustrate XML request and response messages.

Example 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 numeric ID, `124`, for the `GROUP NAME`, rather than the string, `URL_STATS`.

Example C–6   Obtaining the URLs of the Most Popular Objects

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"/>
   <ENTRY NAME="TYPE" VALUE="CACHED"/>
   <ENTRY NAME="REASON" VALUE="BY_CACHING_RULE"/>
```

```
   <ENTRY NAME="COMPRESSED" VALUE="NO"/>
 </GROUP>
.
.
.
</GROUP>
</WCSTATS>
```

Example C–7 shows the request and response messages that retrieve the number of objects recently invalidated for each second. The example uses numeric ID, 107, for the GROUP NAME, rather than the string, INVALIDATED_OBJECTS.

Example 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>
```

Example C–8 shows the request and response messages that retrieve all statistics for the group INVALIDATED_OBJECTS. The example uses numeric ID, 107, for the GROUP NAME, rather than the string, INVALIDATED_OBJECTS. Because the request contains only the GROUP element, OracleAS Web Cache returns all statistics in the group.

Example 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>
```

Example C–9 shows the request and response messages that retrieve TOTAL_SINCE_
START statistics for the group HTTP_REQUESTS for a specific site and for all unnamed
sites.

Example C–9   Obtaining Statistics for a Specific Site and Unnamed Sites

The following code shows the request:

```
<?xml version="1.0"?>
<!DOCTYPE WCSTATS SYSTEM "internal:///wcstats.dtd">

<WCSTATS  DTD_VERSION="1.0">
<
<GROUP NAME="HTTP_REQUESTS">
  <!-- Get statistics for one site. -->
  <PARAM NAME="SITE_NAME" VALUE="myhost:7787"/>
  <ENTRY NAME="TOTAL_SINCE_START"/>
</GROUP>
<GROUP NAME="HTTP_REQUESTS">
  <!-- Get statistics for unnamed sites. -->
  <PARAM NAME="SITE_NAME" VALUE="0"/>
  <ENTRY NAME="TOTAL_SINCE_START"/>
</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="126" NAME="HTTP_REQUESTS">
  <PARAM NAME="SITE_NAME" VALUE="myhost:7787"/>
  <ENTRY NAME="TOTAL_SINCE_START" VALUE="123"/>
</GROUP>

<GROUP NAME="126" NAME="HTTP_REQUESTS">
  <PARAM NAME="SITE_NAME" VALUE="0"/>
  <ENTRY NAME="TOTAL_SINCE_START" VALUE="678"/>
</GROUP>

</WCSTATS>
```

Complete Statistics Template

Example C–10 shows a complete template for all elements in the statistics DTD.

Example C–10   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="CACHE_INFO">
  <ENTRY NAME="MAX_CACHE_SIZE" VALUE=""/>
  <ENTRY NAME="ACTION_LIMIT_SIZE" VALUE=""/>
  <ENTRY NAME="ALLOCATED_MEM_SIZE" VALUE=""/>
  <ENTRY NAME="VERSION" VALUE=""/>
</GROUP>

<GROUP NAME="CACHEABILITY_RULES">
  <GROUP NAME="RULE">
    <ENTRY NAME="INDEX" VALUE=""/>
    <ENTRY NAME="EXPRESSION_TYPE" VALUE="PATH_PREFIX|FILE_EXT|REGEXP"/>
    <ENTRY NAME="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>

<!-- 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">
  <PARAM NAME="SITE" VALUE="hostname:port"/>
  <PARAM NAME="ID" VALUE="n"/>
  <ENTRY NAME="CURRENT" VALUE=""/>
  <ENTRY NAME="MAX_SINCE_START" VALUE=""/>
</GROUP>

<GROUP NAME="CACHE_REDIRECT_DOC_COUNT">
  <PARAM NAME="SITE" VALUE="hostname:port"/>
  <PARAM NAME="ID" VALUE="n"/>
  <ENTRY NAME="CURRENT" VALUE=""/>
  <ENTRY NAME="MAX_SINCE_START" VALUE=""/>
</GROUP>

<!-- Timed Runtime Statistics -->
<GROUP NAME="BYTES_SERVED">
  <PARAM NAME="SITE" VALUE="hostname:port"/>
  <PARAM NAME="ID" VALUE="n"/>
  <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">
  <PARAM NAME="SITE" VALUE="hostname:port"/>
  <PARAM NAME="ID" VALUE="n"/>
  <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">
  <PARAM NAME="SITE" VALUE="hostname:port"/>
  <PARAM NAME="ID" VALUE="n"/>
  <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">
  <PARAM NAME="SITE" VALUE="hostname:port"/>
```

```
              <PARAM NAME="ID" VALUE="n"/>
              <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">
              <PARAM NAME="SITE" VALUE="hostname:port"/>
              <PARAM NAME="ID" VALUE="n"/>
              <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">
              <PARAM NAME="SITE" VALUE="hostname:port"/>
              <PARAM NAME="ID" VALUE="n"/>
              <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">
              <PARAM NAME="SITE" VALUE="hostname:port"/>
              <PARAM NAME="ID" VALUE="n"/>
              <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">
    <PARAM NAME="SITE" VALUE="hostname:port"/>
    <PARAM NAME="ID" VALUE="n"/>
    <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">
      <PARAM NAME="SITE" VALUE="hostname:port"/>
      <PARAM NAME="ID" VALUE="n"/>
      <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">
      <PARAM NAME="SITE" VALUE="hostname:port"/>
      <PARAM NAME="ID" VALUE="n"/>
      <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">
      <PARAM NAME="SITE" VALUE="hostname:port"/>
      <PARAM NAME="ID" VALUE="n"/>
      <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">
      <PARAM NAME="SITE" VALUE="hostname:port"/>
      <PARAM NAME="ID" VALUE="n"/>
      <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 NAME="ESI_UNCAUGHT_EXCEPTIONS">
                  <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="ESI_DEFAULT_FRAGMENT_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>

              <!-- Multisite Statistics -->
              <GROUP NAME="SITE_LIST">
                <GROUP NAME="SITE">
                  <ENTRY NAME="NAME" VALUE=""/>
                  <ENTRY NAME="ID" VALUE=""/>
                </GROUP>
              </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="TYPE="CACHED|NOT_CACHED|ALL"/>
    <PARAM NAME="OBJECT_COUNT" VALUE="n"/>
    <GROUP NAME="URL">
        <ENTRY NAME="NAME" VALUE=""/>
        <ENTRY NAME="SCORE" VALUE=""/>
        <ENTRY NAME="CACHABILITYRULE" VALUE=""/>
        <ENTRY NAME="SIZE" VALUE=""/>
        <ENTRY NAME="TYPE" VALUE=""/>
        <ENTRY NAME="REASON" VALUE=""/>
        <ENTRY NAME="COMPRESSED" VALUE=""/>
    </GROUP>
    .
    .
    .
</GROUP>

<!-- Cache Reason Statistics -->
<GROUP NAME="CACHE_REASON">
    <ENTRY NAME="REASON" VALUE=""/>
</GROUP>

</WCSTATS>
```

D

Caching with Third-Party Application Web Servers

This appendix discusses how to configure OracleAS Web Cache with third-party application Web servers.

This appendix contains these topics:

- Overview of Third-Party Application Servers
- BEA WebLogic Server 9.0
- IBM WebSphere Application Server, Version 6.0
- Apache Tomcat, Version 4.1
- Microsoft IIS 5.0

Notes:

- While this appendix describes how OracleAS Web Cache works with four specific kinds of servers, OracleAS 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 OracleAS Web Cache. Refer to the third-party application Web server documentation for information about designing applications.

Overview of Third-Party Application Servers

Because OracleAS Web Cache is transparent to the application Web server, the application Web server treats HTTP requests from OracleAS 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 OracleAS Web Cache as an HTTP message.

Because OracleAS 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 OracleAS 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
- Caching Rules and Expiration Rules

Web Site Configuration

You configure OracleAS Web Cache to communicate with a third-party application Web server the same way you do with Oracle HTTP Server, by providing the host name and the listening port number. Table D–1 shows the default values for the listening ports for the products discussed in this appendix.

Table D–1   Third-Party Application Web Server Default Listening Ports

Application Web Server	Port
BEA WebLogic Server 9.0	7001
IBM WebSphere Application Server, Version 6.0	80
Apache Tomcat, Version 4.1	8080
Microsoft IIS 6.0	80

To configure OracleAS Web Cache to communicate with a third-party application Web server, perform the following tasks:

1. "Task 6: Configure OracleAS Web Cache with Listening Ports for Client Requests" on page 8-19 to change OracleAS Web Cache port settings

2. "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25 to configure application Web server settings

3. "Task 10: Configure Web Site Settings" on page 8-27 to configure Web site 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 choose to cache or not to cache content for the following:

- Static objects
- Multiple-version objects 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 objects or invalidate objects at any time.

> **See Also:**   Chapter 12, "Creating Caching Rules"

BEA WebLogic Server 9.0

The WebLogic Server installation includes a number of Java Server Pages (JSP), Java servlets, and Enterprise JavaBeans (EJB) examples. This section explains how to configure OracleAS Web Cache to cache the JSP - Simple Tag application.

WebLogic SimpleTag JSP

You can use `SimpleTag.jsp` to demonstrate how OracleAS Web Cache caches full-page dynamic content.

To cache `SimpleTag.jsp`:

1. Ensure that OracleAS Web Cache has been configured to communicate with the WebLogic Application Server, as described in "Web Site Configuration" on page D-2.

2. Start the WebLogic Server, and then access the sample applications page with the following URL:

   `http://hostname:7001/examplesWebApp/index.jsp`

3. Run the JSP - Simple Tags sample application. Access the following URL:

   `http://hostname:7001/jsp_simpleTag/SimpleTag.jsp`

   The sample JSP application displays in the browser window with the BEA icon and a table fetched from the database.

4. Create a caching rule for the JSP - Simple Tag output, as described in "Configuring Caching Rules and Rule Association" on page 12-7.

   When creating the caching rule, configure the following in the Edit/Add Caching, Personalization, and Compression Rule dialog box:

   a. In the **URL Expression** field, enter `/jsp_simpleTag/SimpleTag.jsp`.

   b. In the **HTTP Method(s)** section, click **GET.**

   c. In the **Caching Policy** section, click **Cache.**

   d. Leave all other defaults in the Edit/Add Caching, Personalization, and Compression Rule dialog box as is.

5. Point the browser to OracleAS Web Cache with following URL:

   `http://web_cache_hostname:7777/jsp_simpleTag/SimpleTag.jsp`

   The output is the same as it was when you accessed `SimpleTag.jsp` directly from the WebLogic Server. This time, OracleAS Web Cache caches the `SimpleTag.jsp` output and serves the request to the browser.

   > **Note:** Port 7777 is the default listening port for OracleAS Web Cache. If you changed the default listening port, use that port number. Check the Ports page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**) or the Listen Ports page in OracleAS Web Cache Manager (**Ports** > **Listen Ports**) to determine the correct port for your configuration.

6. View the contents of the cache, as described in "Listing Popular Requests and Cache Contents" on page 15-1, to ensure that `SimpleTag.jsp` is cached.

   From this point on, anytime a browser accesses `SimpleTag.jsp`, the response will be served from OracleAS Web Cache.

`SimpleTag.jsp` 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 JSP.

Use it to demonstrate how OracleAS Web Cache caches pages with **session-encoded URLs**.

To cache `SimpleTag.jsp` for session-encoded URLs:

1. Configure the browser not to accept cookies.

   This is required in order to use session-encoded URLs in this example.

2. Start the WebLogic Server, and then access the sample applications page with the following URL:

   `http://hostname:7001/examplesWebApp/index.jsp`

3. Run the JSP - Simple Tags sample application. Access the following URL:

   `http://hostname:7001/jsp_simpleTag/SimpleTag.jsp`

   The sample JSP application displays in the browser window with the BEA icon and a table fetched from the database.

4. Create an expiration rule, as described in "Configuring Expiration Policies" on page 12-14.

   In the Create Expiration Policy dialog box, perform the following steps:

   a. In the **Expire** section, specify that the output expire 60 seconds after cache entry.

   b. In the **After Expiration** section, select **Remove immediately.**

5. Create a session-caching policy, as described in "Configuring Session or Personalized Attribute Caching Policies" on page 12-19.

   When configuring a session-caching policy for `SimpleTag.jsp`, perform the following steps:

   a. In the Edit/Add Session Definition dialog box:

      * In the **Session Name** field, enter `BEASession`.

      * In the **Cookie Name** field, enter `JSESSIONID`.

        `JSESSIONID` is the default cookie name used by the WebLogic Server.

      * In the **URL or POST body parameter** field, enter `jsessionid`.

   b. In the Add Session Caching Policy dialog box:

      * From the **Please select a session** list, select `BEASession`.

      * Select **YES** for prompt 1.

      * Select **YES** for prompt 2.

      * Select **NO** for prompt 3.

   c. Modify the caching rule for `SimpleTag.jsp`.

      In **Session Caching Policies** section of the Edit/Add Caching, Personalization, and Compression Rule dialog box, select **BEASession**, **Cache with session**, and **Cache without session.**

6. Point the browser to OracleAS Web Cache with the following URL:

   `http://web_cache_hostname:7777/jsp_simpleTag/SimpleTag.jsp`

   When the page is refreshed or reloaded, OracleAS Web Cache serves the content, and the request never goes to the WebLogic Server.

7. View the contents of the cache, as described in "Listing Popular Requests and Cache Contents" on page 15-1, to ensure that `SimpleTag.jsp` is cached.

IBM WebSphere Application Server, Version 6.0

When OracleAS Web Cache fetches static content from IBM Websphere Application server, the IBM Websphere Application Server sends a `content="ESI/1.0+"` directive in the `Surrogate-Control` response header in the response to the OracleAS Web Cache `Surrogate-Capability: orcl="ESI/1.0"` request. If OracleAS Web Cache is deployed as a caching solution, this difference in the control directive value may result in undefined Web application behavior.

To resolve this problem:

1. In the WebSphere Application Server administrative console, navigate to **Servers** > **Application servers**.

   The Application servers page appears.

2. In the Application servers page appears, select the `server1` application server.

   `server1` is the server name when IBM Websphere Application Server is installed with default options. If you specified a different name, select that name instead.

3. In the **Server Infrastructure** section of the **Configuration** tab, select **Java and Process Management**, and then **Process Definition**.

4. In the **Additional Properties** section, select **Java Virtual Machine**, and then **Custom Properties**.

5. Click **New** to create a new entry.

6. In the **Name** field, enter `com.ibm.servlet.file.esi.control`.

7. In the **Value** field, enter `max-age=300, cacheid="URL", content="ESI/1.0"`.

8. Click **Apply**, then save and restart WebSphere Application Server.

The WebSphere Application Server installation includes a number of JSP, Java servlets, and EJB examples. This section explains how to configure OracleAS Web Cache to cache the following content:

- WebSphere Snoop Servlet

- WebSphere Calendar Creator JSP

WebSphere Snoop Servlet

The `snoop` servlet shows getting and using request information, headers, and parameters sent by the browser. Use it to demonstrate how OracleAS Web Cache caches full-page dynamic content.

To cache the `snoop` servlet:

1. Ensure that OracleAS Web Cache has been configured to communicate with the WebSphere Application Server, as described in "Web Site Configuration" on page D-2.

2. Start the WebSphere Application Server, and then access the following URL:

   `http://hostname/snoop`

Notice that request information, headers, and parameters sent by your browser display.

3. Create a caching rule for the `snoop` output, as described in "Configuring Caching Rules and Rule Association" on page 12-7.

When creating the caching rule for the `snoop` output, configure the following in the Edit/Add Caching, Personalization, and Compression Rule dialog box:

   a. In the **URL Expression** field, enter `/snoop`.

   b. In the **HTTP Method(s)** section, click **GET.**

   c. In the **Caching Policy** section, click **Cache.**

   d. Leave all other defaults in the Edit/Add Caching, Personalization, and Compression Rule dialog box as is.

4. Point the browser to OracleAS Web Cache with following URL:

   `http://web_cache_hostname:7777/snoop`

   The output is the same as it was when you accessed `snoop` directly from the WebSphere Application Server. This time, OracleAS Web Cache caches the `snoop` output and serves the response to the browser.

   > **Note:** Port 7777 is the default listening port for OracleAS Web Cache. If you changed the default listening port, use that port number. Check the Ports page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**) or the Listen Ports page in OracleAS Web Cache Manager (**Ports** > **Listen Ports**) to determine the correct port for your configuration.

5. View the contents of the cache, as described in "Listing Popular Requests and Cache Contents" on page 15-1, 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 Calendar Creator JSP

The `Calendar` JSP generates a calendar based on user input. The example is not a pre-deployed WebSphere example like the `snoop` servlet. To find this example, you need to install Technology Samples, as mentioned in the documentation under IBM WebSphere Application Server samples gallery. Use this JSP to demonstrate how OracleAS Web Cache caches pages with session cookies.

To cache `SimpleTag.jsp` for session-encoded URLs:

1. Start the WebSphere Application Server, set the browser to accept cookies, and then access the following URL:

   `http://hostname/TechnologySamples/Calendar`

   Notice that the page displays a form asking for inputs on month, year, and other preferences to create a calendar. To use the application:

   a. Enter some values, and then click **Continue**.

   b. Enter some values in the **Day** and **Memo** fields, and then click **Add Memo**.

   c. Click **Generate Calendar**.

2. Create an expiration rule, as described in "Configuring Expiration Policies" on page 12-14.

In the Create Expiration Policy dialog box, perform the following steps:

a. In the **Expire** section, specify that the output expire 60 seconds after cache entry.

b. In the **After Expiration** section, select **Remove immediately.**

3. Create a session-caching policy, as described in "Configuring Session or Personalized Attribute Caching Policies" on page 12-19.

When configuring a session-caching policy, perform the following steps:

a. In the Edit/Add Session Definition dialog box:

* In the **Session Name** field, enter `IBMSession`.

* In the **Cookie Name** field, enter `JSESSIONID`.

   `JSESSIONID` is the default cookie name used by the WebSphere Application Server.

* In the **URL or Post body parameter** field, enter `jsessionid`.

b. In the Add Session Caching Policy dialog box:

* From the **Please select a session** 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 `Calendar`, as described in "Configuring Caching Rules and Rule Association" on page 12-7.

When creating the caching rule for the `Calendar` output, configure the following in the Edit/Add Caching, Personalization, and Compression Rule dialog box:

* In the **URL Expression** field, enter `/TechnologySamples/Calendar`.

* In the **HTTP Method(s)** section, click **GET.**

* In the **Caching Policy** section, click **Cache.**

* From the **Expiration Policy** list, select Expire: 60 seconds in cache. After: remove immediately.

* In the **Session Caching Policies** section, select **IBMSession**, **Cache with session**, and **Cache without session.**

* Leave all other defaults in the Edit/Add Caching, Personalization, and Compression Rule dialog box as is.

4. Point the browser to OracleAS Web Cache with the following URL:

`http://web_cache_hostname:7777/TechnologySamples/Calendar`

The output is the same as when you access `Calendar` directly from WebSphere Application Server. This time, OracleAS Web Cache caches the `Calendar` output.

> **Note:** Port 7777 is the default listening port for OracleAS Web Cache.
> If you changed the default listening port, use that port number. Check
> the Ports page of Application Server Control Console (**Web Cache
> Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**)
> or the Listen Ports page in OracleAS Web Cache Manager (**Ports** >
> **Listen Ports**) to determine the correct port for your configuration.

5. View the contents of the cache, as described in "Listing Popular Requests and
Cache Contents" on page 15-1, to ensure that `Calendar` 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, OracleAS Web Cache
expires the cached content after 60 seconds and reflects the content the next time
the user requests the page.

After deploying OracleAS Web Cache, if the browser displays a HTTP `404 Page
not found error`, perform the following steps:

1. In the WebSphere Server *WAS_home*`/config/cells/plugin-cfg.xml` file, add
`<VirtualHost Name="*:7777"/>`.

2. In the WebSphere Application Server administrative console, navigate to
Environment > **Virtual Hosts**.

3. Follow the prompts to add a new virtual host `7777`.

> **Note:** Port 7777 is the default listening port for OracleAS Web Cache.
> If you changed the default listening port, use that port number. Check
> the Ports page of Application Server Control Console (**Web Cache
> Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**)
> or the Listen Ports page in OracleAS Web Cache Manager (**Ports** >
> **Listen Ports**) to determine the correct port for your configuration.

Apache Tomcat, Version 4.1

Apache Tomcat, Version 4.1 is a servlet container. It is included with the Apache
Jakarta Project. The Apache Tomcat installation includes a number of JSP and Java
servlet examples. This section explains how to configure OracleAS Web Cache to cache
the following content:

- Apache Tomcat Snoop JSP

- Apache Tomcat Session Servlet

Follow the instructions enclosed within the Apache Tomcat binary for installation.
Apache Tomcat requires the Java Development Kit (JDK).

> **See Also:**
>
> - http://jakarta.apache.org/tomcat/ for further
>   information about Apache Tomcat
>
> - http://java.sun.com for further information about
>   downloading and installing JDK

Apache Tomcat Snoop JSP

`snoop.jsp` shows getting and using request information, headers, and parameters sent by the browser. Use it to demonstrate how OracleAS Web Cache caches full-page dynamic content.

To start, perform the following steps:

1. Ensure that OracleAS Web Cache has been configured to communicate with the Apache Tomcat server, as described in "Web Site Configuration" on page D-2.

2. Start the Apache Tomcat server, and then access the following URL:

   ```
   http://hostname/examples/jsp/snp/snoop.jsp
   ```

   Notice that request information, headers, and parameters sent by your browser display.

To cache this content:

1. Create a caching rule for the `snoop` output, as described in "Configuring Caching Rules and Rule Association" on page 12-7.

   When creating the caching rule for the `snoop` output, configure the following in the Edit/Add Caching, Personalization, and Compression Rule dialog box:

   a. In the **URL Expression** field, enter `/examples/jsp/snp/snoop.jsp`.

   b. In the **HTTP Method(s)** section, click **GET.**

   c. In the **Caching Policy** section, click **Cache.**

   d. Leave all other defaults in the Edit/Add Caching, Personalization, and Compression Rule dialog box as is.

2. Point the browser to the OracleAS Web Cache with following URL:

   ```
   http://web_cache_hostname:7777/examples/jsp/snp/snoop.jsp
   ```

   The output is the same as it was when you accessed `snoop` directly from Apache Tomcat. This time, OracleAS Web Cache caches the `snoop` output and serves the response to the browser.

   > **Note:** Port 7777 is the default listening port for OracleAS Web Cache. If you changed the default listening port, use that port number. Check the Ports page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**) or the Listen Ports page in OracleAS Web Cache Manager (**Ports** > **Listen Ports**) to determine the correct port for your configuration.

3. View the contents of the cache, as described in "Listing Popular Requests and Cache Contents" on page 15-1, to ensure that `snoop` is cached.

   When you reload the page, you should notice that the cached response appears faster than when you access Apache Tomcat directly.

Apache Tomcat Session Servlet

The `SessionServlet` 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 OracleAS Web Cache caches pages with session-encoded URLs.

This servlet may not be included in the Apache Tomcat binary. You can find this example on the Web, or you can use code for the servlet from Example D–1.

Example D–1   Apache Tomcat Binary

```
/*
 * @(#)SessionServlet.java     1.5 1.5
 *
 * Copyright (c) 1996-1998 Sun Microsystems, Inc. All Rights Reserved.
 *
 * This software is the confidential and proprietary information of Sun
 * Microsystems, Inc. ("Confidential Information").  You shall not
 * disclose such Confidential Information and shall use it only in
 * accordance with the terms of the license agreement you entered into
 * with Sun.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THE
 * SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
 * PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR ANY DAMAGES
 * SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING
 * THIS SOFTWARE OR ITS DERIVATIVES.
 *
 * CopyrightVersion 1.0
 */

package sunexamples;

import java.io.*;
import java.util.Enumeration;

import javax.servlet.*;
import javax.servlet.http.*;


/**
 * This is a simple example of an HTTP Servlet that uses the HttpSession
 * class
 *
 * Note that in order to guarantee that session response headers are
 * set correctly, the session must be retrieved before any output is
 * sent to the client.
 */
public class SessionServlet extends HttpServlet {

    public void doGet (HttpServletRequest req, HttpServletResponse res)
      throws ServletException, IOException
      {

        //Get the session object
        HttpSession session = req.getSession(true);

        //Get the output stream
        ServletOutputStream out = res.getOutputStream();

        res.setContentType("text/html");

        out.println("<HEAD><TITLE> SessionServlet Output " +
                "</TITLE></HEAD><BODY>");
        out.println("<h1> SessionServlet Output </h1>");
```

```
        //Here's the meat
        Integer ival = (Integer) session.getValue("sessiontest.counter");
        if (ival==null) ival = new Integer(1);
        else ival = new Integer(ival.intValue() + 1);
        session.putValue("sessiontest.counter", ival);

        out.println("You have hit this page <b>" + ival + "</b> times.<p>");

        // encodeURL Encodes the specified URL by including the session ID in it
        // if cookies are not turned on or not supported by the browser
        out.println("Click <a href=" + res.encodeURL("/session.html") +
                ">here</a>");
        out.println(" to ensure that session tracking is working even if" +
                " cookies aren't supported.<br>");
        out.println(" Note that by default URL rewriting is not enabled due" +
                " to it's expensive overhead.");
        out.println("<p>");

        out.println("<h3>Request and Session Data:</h3>");
        out.println("Session ID in Request: " + req.getRequestedSessionId());
        out.println("<br>Session ID in Request from Cookie: " +
                        req.isRequestedSessionIdFromCookie());
        out.println("<br>Session ID in Request from URL: " +
                        req.isRequestedSessionIdFromURL());
        out.println("<br>Valid Session ID: " +
                        req.isRequestedSessionIdValid());
        out.println("<h3>Session Data:</h3>");
        out.println("New Session: " + session.isNew());
        out.println("<br>Session ID: " + session.getId());
        out.println("<br>Creation Time: " + session.getCreationTime());
        out.println("<br>Last Accessed Time: " +
                            session.getLastAccessedTime());
        out.println("<br><a href=\"/examples/simple_servlets\">Up</a>");
        out.println("</BODY>");
        out.close();
    }

    public String getServletInfo() {
        return "A simple session servlet";
    }
}
```

To start, perform the following steps:

1. Compile the `SessionServlet.java` file in the Apache Tomcat environment.

2. Copy the `SessionServlet.class` to the `/examples/servlets/` directory where other servlet examples may reside.

3. Ensure that OracleAS Web Cache has been configured to communicate with the Apache Tomcat, as described in "Web Site Configuration" on page D-2.

4. Configure the browser not to accept cookies.

   This is required in order to use session-encoded URLs in this example.

5. Start Apache Tomcat and access the following URL:

   ```
   http://hostname/examples/servlets/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 Policies" on page 12-14.

   In the Create Expiration Policy dialog box, perform the following steps:

   a. In the **Expire** section, specify that the output expire 60 seconds after cache entry.

   b. In the **After Expiration** section, select **Remove immediately.**

2. Create a session-caching policy, as described in "Configuring Session or Personalized Attribute Caching Policies" on page 12-19.

   When configuring a session-caching policy for the Session servlet, perform the following steps:

   a. In the Edit/Add Session Definition dialog box:

      * In the **Session Name** field, enter ApacheSession.

      * In the **Cookie Name** field, enter JSESSION.

      * In the **URL or Post body parameter** field, enter jsessionid.

   b. In the Add Session Caching Policy dialog box:

      * From the **Please select a session** list, select ApacheSession.

      * 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 and Rule Association" on page 12-7.

      When creating the caching rule for the Session servlet output, configure the following in the Edit/Add Caching, Personalization, and Compression Rule dialog box:

      * In the **URL Expression** field, enter /examples/servlets/SessionServlet.

      * In the **HTTP Method(s)** section, click **GET.**

      * In the **Caching Policy** section, click **Cache.**

      * From the **Expiration Policy** list, select Expire: 60 seconds in cache. After: remove immediately.

      * In the **Session Caching Policies** section, select **ApacheSession**, **Cache with session**, and **Cache without session.**

      * Leave all other defaults in the Edit/Add Caching, Personalization, and Compression Rule dialog box as is.

3. Point the browser to OracleAS Web Cache with the following URL:

   ```
   http://web_cache_hostname:7777/examples/servlets/SessionServlet
   ```

The output is the same as it was when you accessed `Session` servlet directly from Apache Tomcat. This time OracleAS Web Cache caches the `Session` servlet output. When the page is refreshed or reloaded, notice that the counter does not increment by one. This is because OracleAS Web Cache serves the content, and the request never goes to the Apache Tomcat.

> **Note:** Port 7777 is the default listening port for OracleAS Web Cache. If you changed the default listening port, use that port number. Check the Ports page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**) or the Listen Ports page in OracleAS Web Cache Manager (**Ports** > **Listen Ports**) to determine the correct port for your configuration.

4. View the contents of the cache, as described in "Listing Popular Requests and Cache Contents" on page 15-1 to ensure that `Session` servlet is cached.

    When you reload the page, notice that the cached response appears faster than when you access the Apache Tomcat server directly.

    Because the expiration rule for this URL is set to 60 seconds, OracleAS 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

> **Note:** If you are using Microsoft ASP.NET and you feel the content is cacheable, use the following recommendations when configuring OracleAS Web Cache:
>
> - Create a caching rule to cache all `.aspx` objects.
>
> - Select an expiration policy based on the HTTP `Expires` header.
>
> - If a session-caching policy is required, use session ID `ASP.NET_SessionId`.

The Microsoft IIS installation includes a number of ASP examples. This section explains how to configure OracleAS Web Cache to cache the following content:

- 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 OracleAS Web Cache caches full-page dynamic content.

To start, perform the following steps:

1. Ensure that OracleAS Web Cache has been configured to communicate with IIS, as described "Web Site Configuration" on page D-2.

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 display.

To cache this content:

1. Create a caching rule for the `ServerVariables_JScript.asp`, as described in "Configuring Caching Rules and Rule Association" on page 12-7, using the following information:

   a. In the **URL Expression** field, enter the following:

      `/IISSamples/sdk/asp/interaction/ServerVariables_JScript.asp`.

   b. In the **HTTP Method(s)** section, click **GET.**

   c. In the **Caching Policy** section, click **Cache.**

   d. Leave all other defaults in the Edit/Add Caching, Personalization, and Compression Rule dialog box as is.

2. Point the browser to OracleAS 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, OracleAS Web Cache caches the `ServerVariables_JScript.asp` output and serves the request to the browser.

   Note: Port 7777 is the default listening port for OracleAS Web Cache. If you changed the default listening port, use that port number.

   ---

3. View the contents of the cache, as described in "Listing Popular Requests and Cache Contents" on page 15-1, 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 OracleAS Web Cache caches pages with session cookies.

To start, perform the following steps:

1. Ensure that OracleAS Web Cache has been configured to communicate with IIS, as described in "Web Site Configuration" on page D-2.

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 Policies" on page 12-14.

   In the Create Expiration Policy dialog box, perform the following steps:

   a. In the **Expire** section, specify that the output expire 60 seconds after cache entry.

   b. In the **After Expiration** section, select **Remove immediately**.

2. Create a session-caching policy for Cookie_Jscript.asp, as described in "Configuring Session or Personalized Attribute Caching Policies" on page 12-19.

   When configuring a session caching policy, perform the following steps:

   a. In the **Session Name** field, enter MSSession.

   b. In the **Cookie Name** field, enter CookieJSCript.

   c. In the Add Session Caching Policy dialog box:

   * From the **Please select a session** 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 and Rule Association" on page 12-7.

   e. When creating the caching rule for the Cookie_JScript.asp output, configure the following in the Edit/Add Caching, Personalization, and Compression dialog box:

   * In the **URL Expression** field, enter /IISSamples/sdk/asp/interaction/Cookie_JScript.asp.

   * In the **HTTP Method(s)** section, click **GET.**

   * In the **Caching Policy** section, click **Cache.**

   * From the **Expiration Policy** list, select Expire: 60 seconds in cache. After: remove immediately.

   * In the **Session Caching Policies** section, select **MSSession**, **Cache with session**, and **Cache without session.**

   * Leave all other defaults in the Edit/Add Caching, Personalization, and Compression Rule dialog box as is.

3. Point the browser to OracleAS 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, OracleAS 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 OracleAS Web Cache serves the cached content, and the request never goes to IIS.

   Note: Port 7777 is the default listening port for OracleAS Web Cache. If you changed the default listening port, use that port number.

4. View the contents of the cache, as described in "Listing Popular Requests and Cache Contents" on page 15-1, 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, OracleAS Web Cache expires the cached content after 60 seconds and reflects the content the next time the user requests the page.

E

Troubleshooting OracleAS Web Cache

This appendix describes common problems that you might encounter when using OracleAS Web Cache and explains how to solve them. It contains the following topics:

- Problems and Solutions
- Diagnosing Top User Issues
- Diagnosing Cache Content Results
- Diagnosing Common Edge Side Includes (ESI) Syntax Errors
- Impact of HTTP Traffic Changes
- Resolving Common NZE Errors
- Need More Help?

Problems and Solutions

This section describes common problems and solutions. It contains the following topics:

- Startup Failures
- Load Issues on OracleAS Web Cache Computer
- Performance Degradation and Memory
- Invalidation Timeouts
- Capacity Issues on Origin Server
- Browsers Not Receiving Complete Responses
- Browser Displaying a Page Not Displayed Error
- End-User Performance Monitoring Issues
- Session Binding Not Working in An OracleAS Forms Services Configuration
- XML Parsing Errors of webcache.xml Appears in Event Viewer

Startup Failures

This section describes workarounds to common startup failures:

- Problem 1: Port Conflicts
- Problem 2: OracleAS Web Cache Manager Inaccessible
- Problem 3: Mismatched Oracle Home Definitions

- Problem 4: Privileged Port Numbers

- Problem 5: More Than 1,024 Connections

- Problem 6: Opening Wallet

- Problem 7: Loading Library Object Files

- Problem 8: Permission Denied Error

Problem 1: Port Conflicts

During configuration, you configure listening ports from which OracleAS Web Cache receives client requests. By default, the port is 7777 for HTTP requests. You can configure a port for HTTPS requests.

You also configure listening ports for administration, invalidation, and statistics monitoring requests. By default, the HTTP ports are 9400, 9401, and 9402, respectively. You also configure the advertised port number from which an **origin server** can receive OracleAS Web Cache requests.

When you start OracleAS Web Cache, a port conflict check is performed. If there is a port conflict, OracleAS Web Cache fails to start and port conflicts are reported to the event log file. The following shows an excerpt of the event log with port conflict event messages:

```
[25/Jul/2005:19:12:40 +0000] [notification 9612] [ecid: -]  OracleAS Web Cache 10g (10.1.2)
[25/Jul/2005:19:12:40 +0000] [notification 9403] [ecid: -] Maximum number of file/socket
descriptors set to 950.
[25/Jul/2005:19:12:40 +0000] [notification 13002] [ecid: -] Maximum allowed incoming
connections are 700
[25/Jul/2005:19:12:40 +0000] [alert 13305] [ecid: -] Failed to assign port 7777: Address
already in use
[25/Jul/2005:19:12:40 +0000] [alert 9707] [ecid: -]  Failed to start the server.
[25/Jul/2005:19:12:40 +0000] [alert 9609] [ecid: -] The server process could not initialize.
[25/Jul/2005:19:12:40 +0000] [notification 9610] [ecid: -]  The server is exiting.
```

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

Solution 1

Typically, OracleAS Web Cache and the origin server ports are in conflict. To resolve port conflicts:

1. Verify the port assigned to OracleAS Web Cache from the Ports page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Ports**) or the Listen Ports page of OracleAS Web Cache Manager (**Ports** > **Listen Ports**).

    If multiple instances of OracleAS Web Cache are running on a multihomed host with multiple IP addresses, a port conflict can occur. In OracleAS Web Cache Manager,

    The IP address for the default HTTP ports is set to * in Application Server Control Console and `ANY` in OracleAS Web Cache Manager. Upon startup, OracleAS Web Cache attempts to bind the ports to all IP addresses. If a port conflict occurs, change * or `ANY` to a specific IP address from the Listen Ports page of OracleAS Web Cache Manager.

2. Verify the host names and ports assigned to the origin servers from the Origin Servers page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Origin Servers**) or OracleAS

Web Cache Manager (**Origin Servers, Sites, and Load Balancing** > **Origin Servers**).

3. Restart OracleAS Web Cache.

> **See Also:** "Starting and Stopping OracleAS Web Cache In an Oracle Application Server Configuration" on page 7-1

If the administration port is in conflict, the admin server process will not start and OracleAS Web Cache Manager will not be accessible. The event log will contain messages that resemble the following output:

```
[25/Jul/2005:19:21:42 +0000] [alert 13305] [ecid: -] Failed to assign port 9400: Address
already in use
[25/Jul/2005:19:21:42 +0000] [alert 9707] [ecid: -]  Failed to start the server.
[25/Jul/2005:19:21:42 +0000] [alert 9609] [ecid: -] The server process could not initialize.
[25/Jul/2005:19:21:42 +0000] [notification 9610] [ecid: -]  The server is exiting.
```

To resolve this port conflict, modify the webcache.xml file. The following shows an excerpt of the webcache.xml file with the line for the administration port shown in boldface:

```
...
  <MULTIPORT>
    <LISTEN IPADDR="ANY" PORT="7777" PORTTYPE="NORM"/>
    <LISTEN IPADDR="ANY" PORT="9400" PORTTYPE="ADMINISTRATION"/>
    <LISTEN IPADDR="ANY" PORT="9401" PORTTYPE="INVALIDATION"/>
    <LISTEN IPADDR="ANY" PORT="9402" PORTTYPE="STATISTICS"/>
  </MULTIPORT>
...
```

Problem 2: OracleAS Web Cache Manager Inaccessible

OracleAS Web Cache Manager does not enforce stringent validation checking. This is especially a problem when the admin server process is shut down after applying invalid configuration changes. In that case, the admin server process will not be able to start up, and the OracleAS Web Cache Manager will become inaccessible. Check the event log file for startup errors or the Event Viewer on Windows.

Solution 2

To solve this problem:

- If you want to retain the configuration changes you made, send the webcache.xml file to Oracle Support to troubleshoot the invalid configuration entry.

  Do not edit this configuration file manually, except in the cases described in this guide, or when directed to do so by Oracle Support. Improper editing of this configuration file may cause problems in OracleAS Web Cache.

- If you want to restore configuration to a previous configuration, run the webcachectl reset command to restore to the previous version of the configuration. The opmnctl utility does not provide a reset command.

Problem 3: Mismatched Oracle Home Definitions

If the definition of Oracle home in the webcache.xml configuration file is different than the definition of Oracle home in your environment, OracleAS Web Cache may fail to start.

On UNIX, you may see the following alert message in the event log:

```
No matching CACHE element found in webcache.xml for current host name
(webcache-host) and ORACLE_HOME (Oracle_home).
```

On Windows, you may see the following message:

```
The description for Event ID ( 1 ) in Source ( Oracle-Web-Cache ) cannot be
found. The local computer may not have the necessary registry information or
message DLL files to display messages from a remote computer. The following
information is part of the event: Cannot open log files because NULL socket
indicates problem.
```

During installation, the Oracle home is written to the `ORACLEHOME` attribute of the `CACHE NAME` element in the `webcache.xml` file. The Oracle home is specified with the `$ORACLE_HOME` environment variable on UNIX and the `ORACLE_HOME` parameter located at `HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE` in the Windows registry.

Solution 3

If there is a mismatch, modify either the `ORACLEHOME` attribute in the `webcache.xml` file or the Oracle home environment setting.

In a cluster environment, there is one `CACHE NAME` element in the `webcache.xml` file for each cluster member. Be sure to modify the correct element. After modification, propagate the configuration, as described in "Task 4: Propagate the Configuration to Cluster Members" on page 10-8.

Problem 4: Privileged Port Numbers

Port numbers less than 1024 are reserved for use by privileged processes on UNIX. If you want to configure OracleAS Web Cache to listen on a port less than 1024, such as on port 80, then the `webcached` executable must run with the root privilege.

If `webcached` is not run as root in a configuration with privileged ports, then the `cache` server process fails to start and messages similar to the following output are written to the event log file:

```
[25/Jul/2005:19:27:04 +0000] [notification 9612] [ecid: -]  OracleAS Web Cache 10g
(10.1.2)
[25/Jul/2005:19:27:04 +0000] [notification 9403] [ecid: -] Maximum number of
file/socket descriptors set to 950.
[25/Jul/2005:19:27:04 +0000] [notification 13002] [ecid: -] Maximum allowed
incoming connections are 700
[25/Jul/2005:19:27:04 +0000] [alert 13305] [ecid: -] Failed to assign port 80:
Permission denied
[25/Jul/2005:19:27:04 +0000] [alert 9707] [ecid: -]  Failed to start the server.
[25/Jul/2005:19:27:04 +0000] [alert 9609] [ecid: -] The server process could not
initialize.
[25/Jul/2005:19:27:04 +0000] [notification 9610] [ecid: -]  The server is exiting.
```

Solution 4

> **See:** "Running webcached with Root Privilege" on page 8-49 for instructions on configuring the `webcached` executable to run as root.

Problem 5: More Than 1,024 Connections

For OracleAS Web Cache to support more than 1,024 connections on UNIX, the `webcached` executable must run with the root privilege.

If `webcached` cannot run with the root privilege, the `cache` server process fails to start and messages that resemble the following output are written to the event log file:

```
[25/Jul/2005:19:30:48 +0000] [notification 9612] [ecid: -]  OracleAS Web Cache 10g
(10.1.2)
[25/Jul/2005:19:30:48 +0000] [alert 9604] [ecid: -] Could not increase the number
of file/socket descriptors to 2250.
[25/Jul/2005:19:30:48 +0000] [alert 9707] [ecid: -]  Failed to start the server.
```

Solution 5

> **See Also:**
>
> - "Running webcached with Root Privilege" on page 8-49 for instructions on configuring the `webcached` executable to run with the root privilege
> - "Connection Limit" on page 8-16 for further information about specifying the maximum connection limit

Problem 6: Opening Wallet

OracleAS Web Cache is unable to open the **wallet**. When OracleAS Web Cache is unable to open a wallet, messages similar to the following output are written to the event log file:

```
[25/Jul/2005:19:35:00 +0000] [warning 11917] [ecid: -] SSL wallet wallet file
/etc/ORACLE/WALLETS/janedoe/ewallet.p12 does not exist
[25/Jul/2005:19:35:00 +0000] [warning 11919] [ecid: -] SSL wallet autologin file
/etc/ORACLE/WALLETS/janedoe/cwallet.sso does not exist. Wallet does not appear to
be autologin wallet.
[25/Jul/2005:19:35:00 +0000] [warning 11922] [ecid: -] Wallet wallet fails to open
at location /etc/ORACLE/WALLETS/janedoe, NZERROR 28759, as user janedoe
[25/Jul/2005:19:35:00 +0000] [alert 9609] [ecid: -] The server process could not
initialize.
[25/Jul/2005:19:35:00 +0000] [notification 9610] [ecid: -]  The server is exiting.
```

Solution 6

To resolve this error, perform the following procedure:

1. Ensure that the wallet directory exists:

    - `/etc/ORACLE/WALLETS/user_name` on UNIX

    - `%USERPROFILE%\ORACLE\WALLETS` on Windows

2. Ensure that the auto-login file `cwallet.sso` and wallet file `ewallet.p12` exist.

    If these files do not exist, then an auto-login wallet does not exist. In this case, create the wallet and enable auto-login.

3. 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 9-1

4. Restart OracleAS Web Cache with the following command:

    ```
    opmnctl restartproc ias-component=WebCache
    ```

5. Recheck the event log for wallet errors.

Problem 7: Loading Library Object Files

If during the installation or application of new a patch an object file becomes corrupted, OracleAS Web Cache fails to start. You may see an error similar to the following:

```
error while loading shared libraries: libclntsh.so.10.1: cannot open shared object
file: No such file or directory
```

Solution

To resolve this error, relink the library

1. Set the `ORACLE_HOME` environment variable.

2. Go to the `$ORACLE_HOME/webcache/lib` directory.

3. Run the following command to relink the library

   ```
   make -f ins_calypso.mk install
   ```

4. If you are running OracleAS Web Cache with the root privilege, run `webcache_setup.sh`.

   > **See Also:** "Running webcached with Root Privilege" on page 8-49 for instructions on configuring the `webcached` executable to run with the root privilege

Problem 8: Permission Denied Error

If OracleAS Web Cache is started by a user that is running under a different group which does not have sufficient privileges to change user groups, a message similar to the following is report to the event log file:

```
Permission denied when setting group ID group_ID
```

The root user is the only user permitted to change the group ID of users.

Solution 8

The solution to this error depends upon one of the following possible causes:

- The `opmnctl` or `webcachectl` user does not match the configured process identity user in the Security page (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Origin Servers**) of Application Server Control Console or the Process Identity page (**Properties** > **Process Identity**) of OracleAS Web Cache Manager.

  If the intent is to use a group ID which is different from the group ID of OracleAS Web Cache, you must configure `webcached` to run as the root privilege using the `setidentity` or `setroot` command of `webcache_setuser.sh`.

  > **See Also:** "Running webcached with Root Privilege" on page 8-49 for instructions on using the `setidentity` or `setroot` command of `webcache_setuser.sh`

- The `opmnctl` or `webcachectl` user is running as the same group ID as OracleAS Web Cache, which is different from the one listed in the event log message.

  In this case, you must modify the group ID to the correct group in the Security page (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Origin Servers**) of Application Server Control Console or the Process Identity page (**Properties** > **Process Identity**) of OracleAS Web Cache Manager. To

determine the correct group, log in to the computer using the `opmnctl` or `webcachectl` user, and then use the UNIX command `id` to determine the correct group ID.

- The `opmnctl` or `webcachectl` user is running as the same group ID as OracleAS Web Cache, which is the same as the one listed in the event log.

  To resolve this error, stop all the Oracle Application Server processes, and verify the group ID. Use the UNIX command `id` to determine the user and group ID. If the group ID matches the group ID in the event log message, then you can start the processes again.

Load Issues on OracleAS Web Cache Computer

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

- The uptime load—the average kernel scheduler queue length—is going to be longer. OracleAS Web Cache increases the average queue length (uptime output) by approximately one.

- The CPU load is still low because the work OracleAS Web Cache performs is minimal.

Performance Degradation and Memory

> **See Also:** *Oracle Application Server Performance Guide* for performance tuning tips

Because OracleAS Web Cache is an in-memory cache, it is best to deploy OracleAS Web Cache on a dedicated computer to minimize paging. Unless the computer is dedicated to run OracleAS Web Cache, ensure the maximum cache size does not exceed 20 percent of the total memory.

This section describes workarounds to invalidation timeouts:

- Problem 1: Paging

- Problem 2: OracleAS Web Cache Using Memory than the Maximum Cache Size

Problem 1: Paging

If the time taken to cache or invalidate objects increases, the computer may be paging. Paging can severely degrade performance.

Solution 1

To configure OracleAS Web Cache to work efficiently on a computer with paging, either deploy OracleAS Web Cache on a dedicated computer or reduce the maximum cache size and maximum cached object size from the Resource Limits and Timeouts page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Resource Limits and Timeouts**) or the Resource Limits page of OracleAS Web Cache Manager (**Properties** > **Resource Limits**).

Problem 2: OracleAS Web Cache Using Memory than the Maximum Cache Size

If OracleAS Web Cache uses more memory than the maximum cache size, the increase may be caused by numerous simultaneous requests for objects that are larger than the maximum cached object size. In this situation, because the objects are not cached, OracleAS Web Cache uses more memory processing the requests and forwarding them to the origin server than it would to cache the objects.

Solution 2

Review access logs to determine if many simultaneous requests for large objects have been made and adjust the size of the maximum cached object size so that those objects are cached. In addition, check that a caching rule or response header specifies that the objects are to be cached.

To modify the maximum cache size or the maximum cached object size, set new limits from the Resource Limits and Timeouts page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Resource Limits and Timeouts**) or the Resource Limits page of OracleAS Web Cache Manager (**Properties** > **Resource Limits**).

Invalidation Timeouts

The **invalidation** feature has a default timeout of 300 seconds for the propagation of invalidation requests in the **cache hierarchy** or **cache cluster** deployments.

> **See Also:** "How OracleAS Web Cache Can Cache Dynamically Generated Content" on page 2-3 for an overview of invalidation propagation

This section describes workarounds to invalidation timeouts:

- Problem 1: Invalidation Timeouts in a Cache Hierarchy
- Problem 2: Invalidation Timeouts in a Cache Cluster

Problem 1: Invalidation Timeouts in a Cache Hierarchy

When the timeout is exceeded in a cache hierarchy, a message similar to the following is written to the event log file of the **remote cache** or **subscriber cache**:

```
[28/Jul/2005:01:10:16 +0000] [debug 11739] [ecid: 25088352784,1] New subscriber
 OracleAS Web Cache with IP '130.35.45.41' port '22000' has been established.
 Total subscriber(s): '1'.
...
[28/Jul/2005:01:12:31 +0000] [notification 11734] [ecid: -]
Invalidation sent to subscriber cache with IP '130.35.45.41' port '22002' has
 returned with response code: 'failed-no response code'
```

Solution 1

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. Save `webcache.xml`.

5. Restart OracleAS Web Cache with the following command:

```
opmnctl restartproc ias-component=WebCache
```

Problem 2: Invalidation Timeouts in a Cache Cluster

When the timeout is exceeded in a cache cluster, a message similar to the following is displayed in the response to the invalidation request:

```
Can't connect to the web cache's invalidation listening port.
```

Solution 2

To resolve this error:

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. Save `webcache.xml`.

5. Restart OracleAS Web Cache with the following command:

```
opmnctl restartproc ias-component=WebCache
```

6. Propagate changes to all cache cluster members and restart the other cache cluster members.

   > **See Also:**

Capacity Issues on Origin Server

Problem

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

```
The application Web server is busy. Possible reach capacity.
```

This error indicates that the application Web server has exceeded the maximum number of concurrent connections.

Solution

To resolve this problem, you can either:

- Increase capacity.

  In the Origin Servers page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Origin Servers**) or

OracleAS Web Cache Manager (**Origin Servers, Sites, and Load Balancing** > **Origin Servers**), check the value of the **Capacity** field. This field provides the currently configured capacity. If the capacity can be adjusted, increase it.

> **See Also:** "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25

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

> **See Also:** Chapter 12, "Creating Caching Rules"

Browsers Not Receiving Complete Responses

The client browser is not receiving the complete response.

Problem

If the actual length of a page is less than value of the `Content-Length` response-header field set by the origin server and sent to a client browser by OracleAS Web Cache, 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`, the browser will not receive the complete page. This problem does not occur for cache hits because OracleAS Web Cache correctly calculates the content length itself for pages stored in the cache.

Solution

For cache misses, there are two workarounds for the improper content-length problem:

- Fix your application to ensure that the value of the `Content-Length` response-header field is correct.

- Configure the browser or client emulator to send HTTP/1.0 requests without the `Connection: keep-alive` request-header field.

Browser Displaying a Page Not Displayed Error

Client browsers return an error saying that a page cannot be displayed.

Problem

Microsoft Internet Explorer has known issues with trying to reuse SSL connections after they have timed out. Due to this limitation, users connecting to a Web site using Internet Explorer 5.5 or later release with the following OracleAS Web Cache configuration conditions, may receive an error saying that the page cannot be displayed:

- OracleAS Web Cache has at least one listener port to set to accept HTTPS client requests

- The network connection keep-alive timeout is set a value more than 0.

When OracleAS Web Cache is configured with these settings, Internet Explorer may send HTTPS requests *after* OracleAS Web Cache has already tried to close the connection. Then, the browser returns an error saying that the page cannot be displayed.

Solution

To correct his problem, you can upgrade all clients to use the correct Microsoft patches. For information about the Internet Explorer problem, its workarounds, and links to updates to Internet Explorer, see the following:

- Internet Explorer 5.5:
  `http://support.microsoft.com/default.aspx?scid=kb;EN-US;q305217`

- Internet Explorer 6:
  `http://support.microsoft.com/default.aspx?kbid=831167`

In configurations with public Web sites, this option may not feasible. For these configurations, the Web site administrator can either enable or disable keep-alive timeouts on HTTPS connections from Internet Explorer in the `webcache.xml` file. By default, OracleAS Web Cache disables keep-alive for HTTPS connections from Internet Explorer.

To re-enable keep-alive connection for HTTPS requests from Internet Explorer, perform the following steps. In a cache cluster, you must perform this procedure for each cluster member:

1. Use a text editor to open the `webcache.xml` file.

2. Locate the `CALYPSONETINFO` element:

   ```
   <CALYPSONETINFO...KEEPALIVE4MSIE_SSL="NO".../>
   ```

3. Modify the value of `KEEPALIVE4MSIE_SSL` attribute to `YES`.

4. Save `webcache.xml`.

5. Restart OracleAS Web Cache with the following command:

   ```
   opmnctl restartproc ias-component=WebCache
   ```

End-User Performance Monitoring Issues

This section describes workarounds to end-user performance monitoring:

- Problem 1: Cookie and JavaScript in Pages Monitored by End-User Performance Monitoring

■ Problem 2: Not Enough Data in Access Logs for End-User Performance Monitoring

Problem 1: Cookie and JavaScript in Pages Monitored by End-User Performance Monitoring

End-user performance monitoring creates an additional cookie and inserts JavaScript into pages. The application may generate cookies and JavaScript that conflict with additional cookies and JavaScript created by end-user performance monitoring.

Solution

If this behavior causes a problem for an application, then disable this feature from the End-User Performance Monitoring page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **End-User Performance Monitoring**) or OracleAS Web Cache Manager (**Logging and Diagnostics** > **End-User Performance Monitoring**).

> **See Also:** "Configuring End-User Performance Monitoring" on page 14-3 for further information about end-user performance monitoring

Problem 2: Not Enough Data in Access Logs for End-User Performance Monitoring

You attempt to gather end-user performance monitoring data and receive an error about the access logs not containing enough data. This error can be the result of a configuration error.

Solution 2

To resolve this error:

1. Check that the End-User Performance Monitoring Format is configured as the logging format for the site.

   > **See Also:** "Configuring Access Logs" on page 15-22

2. Ensure end-user performance monitoring is configured for the site.

   > **See Also:** "Enabling End-User Performance Monitoring" on page 14-4

3. Check the site-to-server mappings.

   > **See Also:** "Task 10: Configure Web Site Settings" on page 8-27

Session Binding Not Working in An OracleAS Forms Services Configuration

OracleAS Web Cache fails to bind a session to an origin server in a configuration that uses OracleAS Forms Services.

Problem

The mechanism for session binding is not set for OC4J-based applications. In releases prior to 10*g* Release 2 (10.1.2), OracleAS Web Cache used an internal tracking mechanism that was not suitable for OracleAS Forms Services.

Solution

Configure the OCJ4-based session binding mechanism:

1. Upgrade to OracleAS Web Cache 10*g* Release 2 (10.1.2).

2. Reconfigure session binding, making sure to select the **OC4J-based** mechanism.

   For all sites enabled for end-user performance monitoring, ensure that all possible host names and associated port numbers used to access the site are specified either in the specification of the site, or as one of the aliases of the site. If a site is accessed using a host name and port that is not defined, end-user performance monitoring is not instrumented for the site.

   > **See Also:**
   >
   > - "Bind a Session to an Origin Server" on page 8-39
   > - *Oracle Application Server Forms Services Deployment Guide* for additional configuration tips

XML Parsing Errors of webcache.xml Appears in Event Viewer

XML parsing errors related to not being able to read the `webcache.xml` file are displayed to the Event Viewer rather than to the screen on Windows.

Diagnosing Top User Issues

Common configuration mistakes include:

- Not mapping sites correctly to origin servers from the Sites page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Sites**) or the Site-to-Server Mapping page of OracleAS Web Cache Manager (**Origin Servers, Sites, and Load Balancing** > **Site-to-Server Mapping**).

  When sites are not mapped, OracleAS Web Cache directs requests to the default Oracle HTTP Server. Browsers may receive an HTTP `500` error code.

  Other site configuration errors include:

  - Not specifying all the site aliases

  - Misuse of the wildcard character *

  - Creating multiple site-to-server mappings for a site with multiple origin servers

    > **See Also:** "Task 10: Configure Web Site Settings" on page 8-27

- Port conflicts

  > **See Also:** "Problem 1: Port Conflicts" on page E-2

- Ping URL

  When configuring the **Ping URL** field from the Origin Servers page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Application** > **Origin Servers**) or OracleAS Web Cache Manager (**Origin Servers, Sites, and Load Balancing** > **Origin Servers**), how you enter the URL depends on the origin server. 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, 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. Ensure the URL is cached.

> **See Also:** "Task 9: Configure Origin Server, Load Balancing, and Failover Settings" on page 8-25

- Running `webcached` with root privilege

  You must configure `webcached` to run with root privilege when privileged port numbers less than 1024, there are more than 1,024 file descriptors, or the current `opmnctl` or `webcachectl` user does not match the configured user in the Security page of Application Server Control Console (**Web Cache Home** page > **Administration** tab > **Properties** > **Web Cache** > **Security**) or the Process Identity page of OracleAS Web Cache Manager (**Properties** > **Process Identity**) of OracleAS Web Cache Manager.

  > **See Also:** "Running webcached with Root Privilege" on page 8-49

Diagnosing Cache Content Results

To diagnose if caching rules are set up to serve wrong or outdated content:

1. Determine the contents of the cache by:

   - Listing the most popular requests, either cached or not cached requests, along with the caching rules associated with cached objects

     Instead of listing the default value of 100, expand the list to see if your object is cached.

   - Listing the contents of the cache

   - Previewing invalidation without invalidating actual content

     > **See Also:**
     >
     > - "Listing Popular Requests" on page 15-1 to generate a list of the URLs of the most popular requests since the cache was last started
     >
     > - "Listing All Contents" on page 15-3 to generate a list of the URLs of the objects currently in the cache
     >
     > - Step 7 in "Oracle Enterprise Manager 10g Application Server Control Console or OracleAS Web Cache Manager to Send Invalidation Requests" on page 13-16 or "Invalidation Preview Request Syntax" on page 13-14 to preview invalidation

2. Enable event logging with a logging level of TRACE. Then, resubmit the request.

   Trace level logging shows whether an object is cached and which caching rule it matches.

   > **See Also:** Chapter 15 for further information about enabling event logging

3. Compare the contents of the cache to the caching rules to determine discrepancies or syntax errors.

   Adjust caching rules by adding or removing rules, adjusting expression type syntax, or changing the precedence of rules.

4. Enable access logging. Then, send an explicit request for the object

By analyzing the access log determine, you can determine if OracleAS Web Cache is serving the object from its cache or is forwarding the request to the origin server.

Diagnosing Common Edge Side Includes (ESI) Syntax Errors

The majority of ESI errors are the result of syntax errors in either the template or fragment pages. By analyzing the ESI output in the event log, you can easily diagnose most ESI syntax errors. To avoid unnecessary reporting in the event log, use a verbosity level of WARNING. It is also useful to display the diagnostic information and event log information in the HTML response body.

In addition, developers using Oracle JDeveloper can use the ESI Servlet Filter extension to run and debug ESI or JESI JSPs.

See Also:

- "Configuring Event Logs" on page 15-6

- "Displaying Diagnostic and Event Log Information in the HTML Body or Server Response-Header Field" on page 15-11

- http://www.oracle.com/technology/products/ias/web_cache/index.html for further information about the ESI Servlet Filter

The following topics describe using the event log and HTML response body to diagnose template and fragment syntax errors:

- Template Syntax Error Example

- Fragment Syntax Error Example

- Fragment Syntax Error with Exception Handling Example

Template Syntax Error Example

Consider a template named `exlusion.html` that contains syntax for a nonexistent ESI tag named `<esi:exclude>`:

```
<html><body>
Simple inclusion test.
<esi:exclude src="/cgi-bin/esi-headers.sh?/esi/fragment1.html"/>
</body></html>
```

The response returned to the browser follows:

```
<HTML><HEAD><TITLE>Unsupported ESI feature</TITLE></HEAD>
<BODY>Some ESI features on this page are not supported.
</BODY></HTML>
```

The following shows an event log excerpt that indicates a problem with the `<esi:exlude>` keyword:

```
[24/Jul/2005:16:02:12 -0500] [detail] [ecid: 25732665668,0] [client: 127.0.0.1]
```

```
[host: www.company.com:80] [url: /cgi-bin/esi-headers.sh?/esi/exclusion.html]
[24/Jul/2005:16:02:12 -0500] [error 12086] [ecid: 25732665668,0] ESI syntax error.
Unrecognized keyword exclude is at line 2.
[24/Jul/2005:16:02:12 -0500] [warning 11064] [ecid: 25732665668,0] ESI object
 www.company.com:80/cgi-bin/esi-headers.sh?/esi/exclusion.html parsing error
```

Fragment Syntax Error Example

Consider a template named `inclusion_exclusion.html` that contains the following syntax for including fragment `exlusion1.html`. Notice that HTML does not contain any ESI exception handling tags or attributes.

```
<html><body>
Simple inclusion test.
<esi:include src="/cgi-bin/esi-headers.sh?/esi/frag_exclusion.html"/>
</body></html>
```

Fragment `frag_exclusion.html` contains syntax for a nonexistent ESI tag named `<esi:exclude>`:

```
<esi:exclude src="/cgi-bin/esi-headers.sh?/esi/fragment1.html"/>
```

The response returned to the browser follows:

```
<HTML><HEAD><TITLE>ESI Processing Exception</TITLE></HEAD>
<BODY>The page caused an ESI processing exception.
</BODY></HTML>
```

The following shows an event log excerpt that indicates a problem with the `<esi:exlude>` keyword. As a result of this error and the fact that the ESI in the template does not specify any alternative fragment to serve, the browser is served an ESI exception.

```
[24/Jul/2005:16:10:40 -0500] [detail] [ecid: 25733186204,0] [client: 127.0.0.1]
[host: www.company.com:80] [url: /cgi-bin/esi-headers.sh?/esi/inclusion_
exclusion.html]
[24/Jul/2005:16:10:40 -0500] [error 12086] [ecid: 25733186204,0] ESI syntax error.
Unrecognized keyword exclude is at line 2.
[24/Jul/2005:16:10:40 -0500] [warning 11064] [ecid: 25733186204,0] ESI object
www.company.com:80/cgi-bin/esi-headers.sh?/esi/frag_exclusion.html parsing error
[24/Jul/2005:16:10:40 -0500] [warning 12009] [ecid: 25733186204,0] Incorrect ESI
fragment exception in ESI template
www.company.com:80/cgi-bin/esi-headers.sh?/esi/inclusion_exclusion.html, fragment
www.company.com:80/cgi-bin/esi-headers.sh?/esi/frag_exclusion.html
[24/Jul/2005:16:10:40 -0500] [error 12012] [ecid: 25733186204,0] No exception
handler is defined in template
www.company.com:80/cgi-bin/esi-headers.sh?/esi/inclusion_exclusion.html:.
[24/Jul/2005:16:10:40 -0500] [error 11368] [ecid: 25733186204,0] ESI exception
error response is returned.
```

Fragment Syntax Error with Exception Handling Example

Consider the same `inclusion_exclusion.html` template that contains the following syntax for including fragment `frag_exclusion.html` or alternative fragment `fragment1.html`. When the `exlusion1.html` fragment specified cannot be fetched, the `fragment1.html` fragment specified with the `alt` attribute is served in its place.

```
<html><body>
Simple inclusion test.
<esi:include src="/cgi-bin/esi-headers.sh?/esi/frag_exclusion.html"
```

```
alt="/esi/fragment1.html"/>
</body></html>
```

Fragment `frag_exclusion.html` contains syntax for a nonexistent ESI tag named `<esi:exclude>`:

```
Simple inclusion succeeded.
<esi:exclude src="/cgi-bin/esi-headers.sh?/esi/fragment1.html"/>
```

Therefore, fragment `fragment1.html` is used instead of `frag_exclusion.html` as the fragment:

```
Simple inclusion succeeded.
```

The response returned to the browser follows:

```
<html><body>
Simple inclusion test. Simple inclusion succeeded.
</body></html>
```

The following shows an event log excerpt that indicates a problem with the `<esi:exlude>` keyword. Because of the exception handling, the browser is served the alternative fragment instead of an ESI exception.

```
[24/Jul/2005:16:14:41 -0500] [detail] [ecid: 25733432444,0] [client: 127.0.0.1]
 [host: www.company.com:80] [url: /cgi-bin/esi-headers.sh?/esi/inclusion_
exclusion.html]
[24/Jul/2005:16:14:41 -0500] [error 12086] [ecid: 25733432444,0] ESI syntax error.
Unrecognized keyword exclude is at line 2.
[24/Jul/2005:16:14:41 -0500] [warning 11064] [ecid: 25733432444,0] ESI object
www.company.com:80/cgi-bin/esi-headers.sh?/esi/frag_exclusion.html parsing error
[24/Jul/2005:16:14:41 -0500] [warning 12009] [ecid: 25733432444,0] Incorrect ESI
 fragment exception in ESI template www.company.com:80/cgi-bin/esi-headers.sh?/
esi/inclusion_exclusion.html, fragment
www.company.com:80/cgi-bin/esi-headers.sh?/esi/frag_exclusion.html
```

In addition to analyzing the event log for the sequence of events, you can also view the diagnostic and event log results in the HTML response. The following shows the HTML response when the string +`wcdebug` is appended is the URL. The template diagnostic information, `TU;max-age=30+60;age=0`, means the following:

- `T` means this page is composed an ESI template.

- `U` means this request resulted in an update of stale object.

- `max-age=30+60` means that the object is to expire in 30 seconds from population and to be removed from the cache 60 seconds from the expiration. This provides a total of 90 seconds from population.

- `age=0` in age means that 0 seconds have passed since population of the cache, meaning there are 30 seconds to expiration and 60 seconds to removal.

The fragment diagnostic information, `FM;max-age=30+0;age=0`, means the following:

- `F` means this page is an ESI fragment.

- `U` means this request resulted in a cache miss.

- `max-age=30+0` means that the object is to expire in 30 seconds from population and to be removed from the cache 0 seconds from the expiration. This provides a total of 30 seconds from population.

■ `age=0` in age means that 0 seconds have passed since population of the cache, meaning there are 30 seconds to expiration and removal.

```
Web Cache Debug Info: Web Cache Debug Info: TU;max-age=30+60;age=0
Simple inclusion test: Web Cache Debug Info: Web Cache Debug Info:
TU;max-age=30+60;age=0
Web Cache Debug Info: FM;max-age=30+0;age=0

[EVENTLOG]
[24/Jul/2005:16:17:23 -0500] [detail] [ecid: 25733598670,0] [client: 127.0.0.1]
[host: www.company.com:80] [url: /cgi-bin/esi-headers.sh?/esi/inclusion_
exclusion.html]
[24/Jul/2005:16:17:23 -0500] [error 12086] [ecid: 25733598670,0] ESI syntax error.
Unrecognized keyword exclude is at line 2.
[24/Jul/2005:16:17:23 -0500] [warning 11064] [ecid: 25733598670,0] ESI object
www.company.com:80/cgi-bin/esi-headers.sh?/esi/frag_exclusion.html parsing error
[24/Jul/2005:16:17:23 -0500] [warning 12009] [ecid: 25733598670,0] Incorrect ESI
 fragment exception in ESI template www.company.com:80/cgi-bin/esi-headers.sh?/
esi/inclusion_exclusion.html, fragment
www.company.com:80/cgi-bin/esi-headers.sh?/esi/frag_exclusion.html
Simple inclusion succeeded.
```

Impact of HTTP Traffic Changes

When OracleAS Web Cache is added to an existing application Web server environment, HTTP traffic changes effect the following aspects of the application:

■ Protocol/Hostname/Port Mapping

To ensure traffic is directed through OracleAS Web Cache, configure all absolute URLs to use the protocol, host name, and port number of OracleAS Web Cache. Also, ensure the `Port` directive in the Oracle HTTP Server `httpd.conf` file specifies the OracleAS Web Cache listening port.

■ SSL Processing

Add certificate management to OracleAS Web Cache, if the connection between the client and OracleAS Web Cache is HTTPS, but the connection between OracleAS Web Cache and the origin server is HTTP.

■ Page Delivery Timing

For compressed pages or pages that requires processing, OracleAS Web Cache waits for an entire page from the origin server before it sends it to the browser.

■ HTTP Protocol

OracleAS Web Cache transparently performs the following:

■ OracleAS Web Cache upgrades and downgrades the protocol version between the origin server and client.

■ For cacheable objects, OracleAS Web Cache sends content to clients with the `Content-Length` response header instead of chunked encoding for the initial request.

■ For cache hits, OracleAS Web Cache overwrites the `Content-Length` response-header field whenever it is different from what the origin server sent. This feature ensures browsers receive full page content.

Resolving Common NZE Errors

Several SSL errors are reported with NZE errors in the event log, as listed in Chapter 17. These errors are described in full in the *Oracle Database Error Messages*. Because this document is not available with the Oracle Application Server documentation library, the most common NZE errors reported in the OracleAS Web Cache event log are listed here to aid with troubleshooting:

NZE-28750: unknown error

> **Cause:** An Oracle security server error occurred of an unspecified type.

> **Action:** Contact Oracle Support. This error should not be visible to applications.

NZE-28862: SSL connection failed

> **Cause:** This error occurred because the peer closed the connection.

> **Action:** Retry the connection.

NZE-28865: SSL connection closed

> **Cause:** The SSL connection closed because of an error in the underlying transport or because the peer process quit unexpectedly.

> **Action:** Retry the connection.

NZE-28867: Integer value is too big.

> **Cause:** The certificate presented had an integer field whose value was too large.

> **Action:** See *Oracle Application Server Security Guide* to find out how to obtain the peer's certificate. Then contact Oracle Support with the peer's certificate chain.

Need More Help?

You can find more solutions on Oracle *MetaLink*, http://metalink.oracle.com. If you do not find a solution for your problem, log a service request.

> **See Also:** *Oracle Application Server Release Notes*, available on the Oracle Technology Network:
> http://www.oracle.com/technology/documentation/index.html

Glossary

access log

A log file that contains information about the HTTP requests sent to OracleAS 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 OracleAS 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 clients requests. The application on the Web server interfaces with the database and performs the job requested by the Web server.

auto-restart

A mechanism that checks if the **cache server process** is running and automatically restarts the `cache` server process if it is not running.

cache cluster

A loosely coupled collection of cooperating OracleAS Web Cache cache instances working together to provide a single logical cache. This configuration is also known as OracleAS Cluster (Web Cache). Cache clusters provide failure detection and failover of caches, increasing the availability of your Web site.

cache cluster member

An instance of OracleAS Web Cache configured with other instances of OracleAS 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 OracleAS Web Cache caches content from another OracleAS Web Cache to a local market. OracleAS 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 hit

An HTTP or HTTPS request that can be served from objects stored in the OracleAS Web Cache cache without going to the **origin server**.

cache miss

An HTTP or HTTPS request that cannot be served from the cache and must be forwarded to an **origin server**.

cache server process

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

capacity

For origin servers, the maximum number of concurrent connections that the **origin server** can accept.

For cache clusters, the absolute capacity for the number of concurrent incoming connections to this cache cluster member from all other cache cluster members, and the relative capacity of the cache cluster member.

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 OracleAS 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 Log Format (CLF)**.

Common Log 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.

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 objects.

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 objects. OracleAS 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 OracleAS 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 OracleAS Web Cache event and error information. 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.

expiration

A function that marks objects as invalid after a certain amount of time in the cache. When objects are marked as invalid and a client requests them, they are either immediately removed and refreshed or refreshed based on when the **origin server** can refresh them.

Extended Log Format (XLF)

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

Extensible Markup Language (XML)

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

failover

When an **origin server** fails, OracleAS Web Cache automatically distributes the load over the remaining origin servers and polls the failed origin server for its current up/down status until it is back online. In a cache cluster environment, OracleAS Web Cache transfers ownership of the content of the failing member to the remaining cluster members.

failure detection

In a cache cluster environment, OracleAS 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.

garbage collection

An OracleAS Web Cache process that removes stale objects based on **popularity** and **validity**.

HTTP protocol

Hypertext Transfer 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

An OracleAS Web Cache function that marks objects as invalid. When objects are marked as invalid and a client requests them, they are removed and then refreshed with new content from the **origin server**. Invalidation keeps the OracleAS Web Cache cache consistent with the content on the origin servers.

invalidation coordinator

In a cache cluster environment, OracleAS 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 7 (L7) switch

A networking switch that provides load balancing functionality at Layer 7 of the **Open Systems Interconnection (OSI)** model—the Application layer. L7 switches base their load balancing decisions on URL content.

load balancer

A mechanism for balancing the load of incoming requests. This mechanism is typically a hardware load balancer in the form of a network switch, such as **Layer 7 (L7) switch**. A hardware load balancer is typically positioned in front of the OracleAS Web Cache server. OracleAS Web Cache can act as a software load balancer for environments where a hardware load balancer is not available.

on-demand content

In a cache cluster environment, on-demand content consists of popular objects 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 among 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.

OPMN

See **Oracle Process Manager and Notification (OPMN) Server**.

Oracle Enterprise Manager

A tool for administering Oracle 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 OracleAS Web Cache

- View performance metrics

Oracle Process Manager and Notification (OPMN) Server

Oracle Process Manager and Notification (OPMN) Server manages Oracle Application Server processes, including Oracle HTTP Server, OC4J, and OracleAS Web Cache processes, and channels all notifications from different components instances to all interested in receiving them. OPMN enables you to administer the OracleAS Web Cache processes, including the **admin server process** and **cache server process**.

OracleAS Web Cache Manager

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

origin server

A server that is either an **application Web server** for internal sites or a **proxy server** for external sites outside a firewall.

OSI

See **Open Systems Interconnection (OSI)**.

owned content

In a cache cluster environment, content that is owned by a particular cache cluster member. OracleAS 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 OracleAS Web Cache to independently cache and manage fragments of HTML objects. A template page is configured with **Edge Side Includes (ESI)** markup tags that tell OracleAS 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 OracleAS Web Cache to assign a queue order to objects. These heuristics determine which objects can be served stale and which objects must be retrieve immediately. While objects with a higher priority are retrieved first, objects with a lower priority are retrieved at a later time.

The queue order of objects is based on the popularity of objects and the validity of objects assigned during invalidation. If the current load and capacity of the **origin server** is not exceeded, then the most popular and least valid objects 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 OracleAS Web Cache to substitute values for personalized attributes based on the information contained within a **cookie** or an **embedded URL parameter**.

popularity

The number of requests for an object since entering the cache and the number of recent requests for the object.

POST body parameter

Parameter information embedded in the `POST` body of objects.

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 OracleAS 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 a source of content for a **provider cache** and a **subscriber cache**.

regular expression

OracleAS 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 OracleAS 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 consistent content.

reverse proxy

A server that appears to be the content server to clients but internally retrieves its objects from other backend origin servers as a proxy. A reverse proxy acts a gateway to the origin servers. It relays requests from outside the firewall to origin servers behind the firewall, and delivers retrieved content back to the client.

round robin

A method of managing server congestion by distributing connection loads across multiple servers. Round robin works on a rotating basis in that the first origin server in the list of configured servers receives the request, then the second origin server receives the second request, and so on.

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, OracleAS Web Cache is able to cache pages for **HTTPS protocol** requests.

selectors

OracleAS Web Cache uses selectors to filter through the caching rules to locate the appropriate rule for the request. Cacheability can be evaluated against the following selectors:

- Expression type
- **URL** expression
- **HTTP request method** of objects
- Embedded URL and **POST body parameter**s
- Body of an HTTP **POST method**

> **See Also:** "Selectors" on page 12-2

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 OracleAS 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 OracleAS 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://www.oracle.com/technology/index.html specifies the location and path a browser will travel to find the main page of the Oracle Technology Network 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 an object.

virtual host site

A site hosted by OracleAS Web Cache. Browsers can request cached content from these sites through OracleAS Web Cache. In addition to caching content, OracleAS 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 available capacity

The percentage of the available **capacity** that the **origin server** can accept.

webcachectl utility

A utility used to start, stop, and restart the admin server process, the cache server process, and the auto-restart process, if OracleAS Web Cache is running in a standalone environment (that is, you installed OracleAS Web Cache from a kit that included only this product; you did not install OracleAS Web Cache as part of an Oracle Application Server installation).

However, beginning with OracleAS Web Cache 10*g* (9.0.4), when OracleAS Web Cache is installed as part of an Oracle Application Server installation, you *must* use OPMN to start, stop, and restart the processes. See **Oracle Process Manager and Notification (OPMN) Server**.

XLF

See **Extended Log Format (XLF)**.

XML

See **Extensible Markup Language (XML)**.

Index

Symbols

" (double quotes) symbol
    regular expression,   13-12, 13-17, 13-19
$ (dollar sign) symbol
    embedded URL parameter,   2-4
    path prefix expression,   12-3, 12-8, 13-9, 13-19
    regular expression,   12-3, 13-20
& (ampersand) symbol
    embedded URL parameter,   2-4
    regular expression,   13-12, 13-17, 13-19
* (asterisk) symbol
    path prefix expression,   12-3, 12-8, 13-9, 13-19
    regular expression,   12-4, 13-20, 14-6
+wcdebug string,   15-13
. (period) symbol
    path prefix expression,   12-3, 12-8, 13-9, 13-19
    regular expression,   12-4, 13-20, 14-6
; (semi-colon) symbol
    embedded URL parameter,   2-4
<!--esi--> tag, Edge Side Includes (ESI),   16-34
> (greater than sign) symbol
    regular expression,   13-12, 13-17, 13-19
? (question mark) symbol
    embedded URL parameter,   2-4
    path prefix expression,   12-3, 12-8, 13-9, 13-19
    regular expression,   12-4, 13-20, 14-6
[ ] (brackets) symbol
    path prefix expression,   12-3, 12-8, 13-9, 13-19
    regular expression,   13-20
\ (backslash) symbol,   14-5
    path prefix expression,   12-3, 12-8, 13-9, 13-19
    regular expression,   12-4, 13-20, 14-6
^ (caret) symbol
    path prefix expression,   12-3, 12-8, 13-9, 13-19
    regular expression,   12-3, 13-20
{ } (braces) symbol
    path prefix expression,   12-3, 12-8, 13-9, 13-19
    regular expression,   13-20
' (single quotes) symbol
    regular expression,   13-12

Numerics

1024 port,   8-20, 8-24, 8-49, E-4, E-14
7001 port,   D-2

7777 port,   8-3, 8-19
7778 port,   8-25
80 port,   8-3, 8-19, D-2
9400 port,   8-3, 8-23
9401 port,   8-3, 8-23
9402 port,   8-3, 8-23, 14-1

A

accelerating Web sites,   5-5
Accept request-header field,   2-6, 15-21
Accept-Charset request-header field,   2-6
Accept-Encoding request-header field,   2-6
Accept-Language request-header field,   2-6
access log fields
    bytes,   15-16
    c-ip,   15-16, 15-17
    cs,   15-16
    cs-bytes,   15-16
    cs-method,   15-16
    cs-uri,   15-16
    cs-uri_stem,   15-16
    cs-uri-query,   15-16
    date,   15-17
    r-time-taken,   15-17
    sc,   15-17
    sc-status,   15-17
    s-ip,   15-17
    time,   15-17
    time-taken,   15-17
    x-auth-id,   15-17
    x-cache,   15-17
    x-cache-detail,   15-18
    x-clf-date,   15-18
    x-cluster,   15-18
    x-conn-abrt,   15-19
    x-cookie,   15-18
    x-date-end,   15-19
    x-date-start,   15-19
    x-ecid,   15-19
    x-esi-info,   15-19
    x-glcookie-set,   15-19
    x-log-id,   15-19
    x-os-name,   15-19
    x-os-timeout,   15-19
    x-protocol,   15-19

T