Oracle® Fusion Middleware

Administrator's Guide for Oracle Web Cache

11*g* Release 1 (11.1.1)

E10143-04

January 2011

ORACLE®

Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache, 11*g* Release 1 (11.1.1)

E10143-04

Primary Author: Deborah Steiner

Contributor: Rick Anderson, Fang Chen, Joseph Errede, Patrick Fry, Hideaki Hayashi, Kurt Heiss, Suresh Kotha, Gary Ling, Rabah Mediouni, Mohamed Sharfudeen, Raymond Pfau, Michael Skarpelos, Parthiban Thilager, Bill Wright, Zhong Xu, Rama Vijjapurapu, Jean Zeng, Naveen Zulpuri

Contents

3 Configuring High Availability Solutions

6 Caching and Compressing Content

7 Invalidating Content

8 Using Diagnostic Features

9 Logging

12 Caching with Third-Party Application Servers

A Troubleshooting Oracle Web Cache

Preface

Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache describes how to use Oracle Web Cache to cache both static and dynamically generated content for at least one **origin server**.

Audience

Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache is intended for Web site administrators who perform the following tasks:

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

To use this guide, become familiar with release 1.0 and 1.1 of the **HTTP protocol**, as well as application server and DNS administration.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. 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.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Related Documents

For more information, see the following documents:

- *Oracle Fusion Middleware 2 Day Administration Guide*
- *Oracle Fusion Middleware Administrator's Guide*
- *Oracle Fusion Middleware Oracle Process Manager and Notification Server Administrator's Guide*
- *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*

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 This Guide?

This preface introduces the new and changed administrative features of Oracle Web Cache that are described in this guide, and provides pointers to additional information.

New Features for Release 11*g*

11*g* Release 1 (11.1.1) includes many new features:

- **Request filtering:** You can configure Oracle Web Cache with request filters to take actions on incoming requests based on certain attributes of the request. An incoming request must pass through configured request filters for a given site before Oracle Web Cache processes it. By configuring request filters, you can prevent malicious code from exploiting software vulnerabilities. For more information, see Chapter 4, "Configuring Request Filtering."
- **MIME Type Match Criteria for Caching Rules:** In addition to specifying the criteria for matching a caching rule to incoming requests based on the URL expression, you can base the match evaluation on the value of the `Content-Type` response header of objects. This feature simplifies the definition of caching rules, reduces the overall number of caching rules, and improves Oracle Web Cache performance. For more information, see Section 6.4 and Section 6.8.1.
- **Invalidation using response headers:** You can enable an origin server to invalidate cached content through an HTTP response header. For more information, see Section 7.4.
- **Request-based logging:** Oracle Web Cache stores every request internally and then writes them in bulk after the request to the event logs. In this way, Oracle Web Cache groups all the requests For more information, see Section 9.1.
- **Oracle Diagnostic Logging (ODL) format for event logs:** Oracle Web Cache supports the ODL format, which provides a common format for all diagnostic messages and log files. For more information, see Section 9.1.
- **Audit logging:** Oracle Web Cache supports the Common Audit Framework for providing a uniform system for administering audits across Oracle Fusion Middleware components. The audit log files generated by Oracle Web Cache processes provide important information that can help you identify and diagnose potential security performance and configuration issues. For more information, see the *Oracle Fusion Middleware Security Guide*.
- **Secure caching:** You can configure Oracle Web Cache to support caching content that is secured by Oracle Single Sign-On authentication with no other authorization requirements. For more information, see Section 5.8.

Part I

Understanding Secure Proxy Caching and Load Balancing

This part presents introductory and conceptual information about Oracle Web Cache. It contains the following chapter:

1 Understanding Reverse Proxying

This chapter provides a general introduction to Oracle Web Cache and its role in providing secure reverse proxying.

This chapter includes the following topics:

1.1 About the Web Tier

The Web tier of a J2EE application server is responsible for interacting with the end users, such as Web browsers primarily in the form of HTTP requests and responses. It is the outermost tier in the HTTP stack, closest to the end user. At the highest level, the Web tier performs four basic tasks:

- Interprets client requests
- Dispatches those requests to an object (for example, an enterprise Java bean) that encapsulates business logic
- Selects the next view for display,
- Generates and delivers the next view

The Web tier receives each incoming HTTP request and invokes the requested business logic operation in the application. Based on the results of the operation and state of the model, the next view is selected to display. The selected view is transmitted to the client for presentation.

Oracle Web Cache is a content-aware server accelerator, or **reverse proxy**, for the Web tier that improves the performance, scalability, and availability of Web sites that run on any Web server or application server, such as Oracle HTTP Server and Oracle WebLogic Server.

Oracle Web Cache is the primary caching mechanism provided with Oracle Fusion Middleware. Caching improves the performance, scalability, and availability of Web sites that run on Oracle Fusion Middleware by storing frequently accessed URLs in memory.

By storing frequently accessed URLs in memory, Oracle Web Cache eliminates the need to repeatedly process requests for those URLs on the application Web server and database tiers. Unlike legacy proxies that handle only static objects, Oracle Web Cache caches both static and dynamically generated content from one or more application Web servers. Because Oracle Web Cache can 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, Oracle Web Cache is also an order of magnitude faster than object caches that run within the application tier.

Because Web Cache is fully compliant with HTTP 1.0 and 1.1 specifications, it can accelerate Web sites that are hosted by any standard Web servers, such as Apache Tomcat and Microsoft IIS. In Oracle Fusion Middleware, Oracle Web Cache resides in front of one or more instances of Oracle HTTP Server. Responses to browser based HTTP requests are directed to the Oracle HTTP Server instance and transmitted through Oracle Web Cache. The Oracle Web Cache instance can handle any Web content transmitted with the standard HTTP protocol.

1.1.1 Reverse Proxying

You can configure Oracle Web Cache as a reverse proxy to origin servers, such as Oracle HTTP Server.

A reverse proxy appears to be the content server to clients but internally retrieves its objects from other back-end origin servers as a proxy. A reverse proxy acts as 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.

Figure 1–1 shows an overview of how reverse proxy Web caching works. Oracle 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 Oracle Web Cache are as follows:

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 Oracle Web Cache server 144.25.190.241.

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

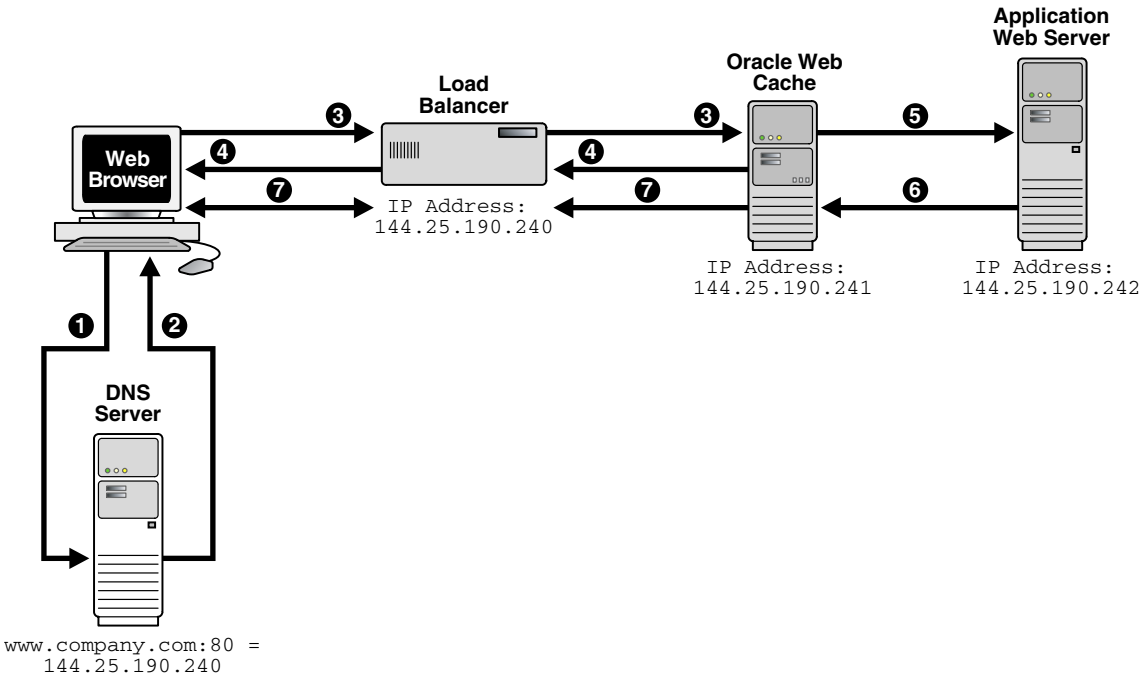
5. If Oracle Web Cache does not have the requested content or the content is stale or invalid, it hands the request off to application Web server 144.25.190.242. This is called a **cache miss**.

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

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

   A page stored in the cache is removed when it becomes invalid or outdated.

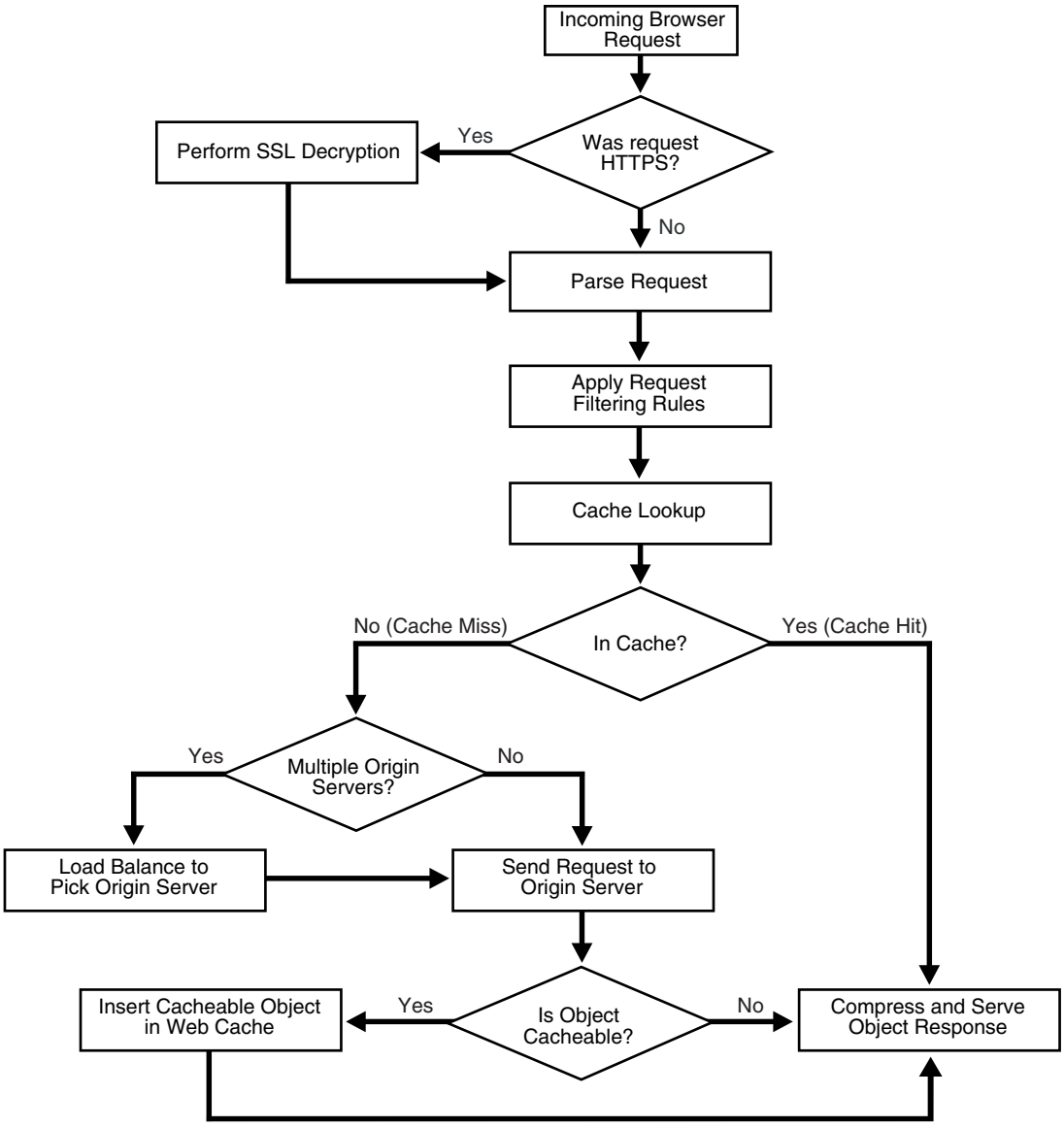
Figure 1–1 Web Server Acceleration



1.2 Request Flow in the Web Tier

Figure 1–2 shows further details of the request flow within the Web tier.

Figure 1–2 Request Flow to Oracle Web Cache within the Web Tier



As shown in Figure 1–2, the following occurs within the Web tier:

1. The incoming browser request is analyzed for the correct HTTP format.
2. The browser request is then further analyzed to determine if it is in HTTPS format:
   a. If the browser request is in HTTPS format, SSL decryption is performed.
   b. If the browser request is not in HTTPS format, the request is parsed.
3. After the request is understood, it is filtered by a set of prescribed filtering rules.
4. A cache lookup is performed to see if the HTTP request was sent previously and is present in the cache.

   If the request is present in the cache, a cache hit, the request is compressed and the content is sent directly to the browser.

   If the request is not present in the cache, a cache miss, then either:

a. The request is sent directly to a single origin server.

b. The request is sent to load-balanced origin servers.

Each load balanced origin server pings each Oracle Web Cache server on a periodic basis to check the status of the cache. The load balancer distributes any incoming requests among cache cluster members. If Oracle Web Cache does not have the requested content or the content is stale or invalid, it hands the request off to the application Web server. The application Web server sends the content to Oracle Web Cache. Oracle Web Cache sends the content to the client and stores a copy of the page in cache.

The proxy server is placed in a less secure zone, the Demilitarized Zone (**DMZ**), instead of the origin server.

Caching rules determine which objects to cache. 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, Oracle Web Cache sends the request to the origin server. This request is a cache miss. Because this URL has an associated caching rule, Oracle Web Cache caches the object for subsequent requests. When Oracle Web Cache receives a second request for the same object, Oracle Web Cache serves the object from its cache to the client. This request is a cache hit.

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

1.2.1 HTTP Traffic Management

You can deploy Oracle Web Cache inside or outside a firewall. Deploying Oracle Web Cache inside a firewall ensures that HTTP traffic enters the DMZ, but only authorized traffic from the application Web servers can directly interact with the database. When deploying Oracle Web Cache outside a firewall, the throughput burden is placed on Oracle Web Cache rather than the firewall. The firewall receives only requests that must go to the application Web servers. This topology requires securing Oracle 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 Oracle Web Cache outside the DMZ.

1.2.2 Request Filtering and Routing

Request filtering checks either the normalized request (for most filter types) or the original raw un-normalized request (for the following format filter rules: null byte, strict encoding, and double encoding). If a match is found on a rule and it is a deny rule, then the request is denied. If the match is for an allow rule, then the request is allowed. For a deny rule, if the rule is in monitor only mode, then the request is logged (to the audit log and the event log), but the request is not denied.

For more information about request filtering, see Chapter 4, "Configuring Request Filtering."

1.2.3 Origin Server Load Balancing and Failover

Origin server load balancing is a feature in which HTTP requests are distributed among origin servers so that no single origin server is overloaded.

Oracle Web Cache supports load balancing and failover detection for application Web servers.

Oracle Web Cache ensures that cache misses are directed to the most available, highest-performing Web server in the server farm. A capacity heuristic guarantees performance and provides surge protection when the application Web server load increases.

For more information about load balancing and failover, see Section 3.1.

1.2.4 Caching

Caching improves the performance, scalability, and availability of Web sites that run on Oracle Fusion Middleware by storing frequently accessed URLs in memory, Oracle Web Cache eliminates the need to repeatedly process requests for those URLs on the application Web server and database tiers. Unlike legacy proxies that handle only static objects, Oracle Web Cache caches both static and dynamically generated content from one or more application Web servers. Because Oracle Web Cache can 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, Oracle Web Cache is also an order of magnitude faster than object caches that run within the application tier.

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

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

- Performance: Running on inexpensive hardware, caching combined with compression can increase the throughput of a Web site by several orders of magnitude. In addition, Oracle 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. See Section 1.2.5 for more information about compression.
- Scalability: In addition to unparalleled throughput, Oracle 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: Oracle 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, Oracle Web Cache offers a simple and inexpensive means of reducing a Web site's cost for each request.
- Developer productivity: Application developers can use Oracle Web Cache to cache content rather than design and develop application-specific caches.

For more information about caching, see Chapter 6, "Caching and Compressing Content."

1.2.5 Compression

Oracle Web Cache can compress both cacheable and non-cacheable objects. You can specify compression settings from either Oracle Enterprise Manager Fusion Middleware Control or the `compress` control directive of the `Surrogate-Control` response-header field. Oracle Web Cache provides compression configuration at both the site and caching-rule level. If you enable compression for a site, then Oracle Web Cache performs automatic compression for that site. Fine tuning of compression settings can be done by configuring individual caching rules.

Oracle Web Cache correctly handles compression of different types of content and different types of browsers. It enables compression automatically for common compressible content types such as HTML, Javascript, or cascading style sheets (CSS). It disables compression automatically where compression either breaks the application in browsers, or does not provide any gain. These files types include GIF, JPEG, and PNG images, or files that are already compressed with utilities like WinZip or GZIP. Similarly, Oracle Web Cache disables compression for Netscape 4 browsers and for some file types for Internet Explorer 5.5 browsers due to known bugs with these browsers.

Because compressed objects are smaller, 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.

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

For more information about compression, see:

- Section 2.11.3 for instructions on configuring compression at the site level
- Section 2.11.3.1 for instructions on disabling compression for all requests
- Section 6.8.1 for instructions on configuring compression for individual caching rules
- Section 6.10 for instructions on configuring the `Surrogate-Control` response-header field

1.2.6 Session Binding

Oracle Web Cache supports sites that use a session ID or **session cookie** to bind user sessions to a given origin server to maintain state for a period. To use the **session binding** feature, the origin server itself must maintain state, that is, it must be stateful. A site 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 Oracle 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. Oracle 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.

For more information about session binding, see Section 3.2.

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

1.3 Compatibility with Oracle Fusion Middleware Components

Table 1–1 describes Oracle Web Cache compatibility with several Oracle Fusion Middleware components. It is not an exhaustive list.

Table 1–1 Compatibility with Other Oracle Fusion Middleware Components

Component	Description
Oracle HTTP Server	In Oracle Fusion Middleware, Oracle Web Cache resides in front of one or more instances of Oracle HTTP Server. Responses to browser based HTTP requests are directed to the Oracle HTTP Server instance and transmitted through Oracle Web Cache. The Oracle Web Cache instance can handle any Web content transmitted with the standard HTTP protocol. **See Also:** *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*
Oracle Business Intelligence Discoverer	Oracle BI Discoverer is closely integrated with Oracle Web Cache to improve Discoverer Viewer's overall scalability, performance, and availability. Oracle BI 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 Oracle BI 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 Forms Services	You can deploy Oracle Web Cache as a load balancer with Oracle Forms Services applications. **See Also:** *Oracle Fusion Middleware Forms Services Deployment Guide*
Oracle Portal	Oracle Web Cache has been closely integrated with Oracle Portal to improve its overall scalability, performance, and availability. Oracle Portal ships with several pre-defined caching and invalidation policies that ensure optimal use of Oracle Web Cache. Oracle Web Cache controls have been built into the Oracle Portal administrative user interface and can also be specified by content providers through the Portal Developer Kit (PDK). **See Also:** *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*

Part II

Basic Administration

This part presents information about performing basic administration tasks for Oracle Web Cache. It contains the following chapters:

2

Getting Started with Administering Oracle Web Cache

This chapter describes how to get started with administering Oracle Web Cache. It discusses the main administration tasks.

This chapter includes the following topics:

2.1 About Oracle Web Cache Management Tools

Oracle stores configuration for Oracle Web Cache in the `webcache.xml` file, located in the following directories:

```
(UNIX) ORACLE_INSTANCE/<instance_name>/config/WebCache/<webcache_name>
(Windows) ORACLE_INSTANCE\<instance_name>\config\WebCache\<webcache_name>
```

Oracle offers two tools for managing Oracle Web Cache:

- Oracle Enterprise Manager Fusion Middleware Control. See Section 2.6.
- Oracle Web Cache Manager. See Section 2.7.

Use these tools rather than edit the `webcache.xml` configuration file, to perform all administrative tasks unless a specific procedure requires you to edit a file. Editing a file may cause the settings to be inconsistent and generate problems.

2.2 About Site Configuration

Oracle Web Cache caches and assembles dynamic content for one or more Web sites. When you configure the following properties for Oracle Web Cache, you apply them to all sites or to a particular site:

- Sessions
- Security
- Request filtering
- Caching rules
- Access logging

You must 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. Oracle 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, Oracle Web Cache caches the same content from either `company.com:80` or `www.company.com:80`.

When configuring a named site, you can enable compression for a site, permitting Oracle Web Cache to perform automatic compression for that site. You can also configure compression for undefined sites. Oracle Web Cache uses the compression setting for undefined sites for client requests that do not match a defined site. If you prefer to disable compression for all requests, see Section 2.11.3.1.

In addition to configuration for named sites and undefined sites, Oracle Web Cache provides configuration for a default site for client requests without host information. When you install Oracle Web Cache, the default site uses the host name and listening port of the computer on which Oracle HTTP Server was installed.

Because Oracle Web Cache resolves a request first to a site definition, and then to the first matching site-to-origin server mapping, the order in which you configure the site definitions is important.

For example, consider site definitions configured in this order:

```
www.company.com:80
www.company.com:80/sales
```

Because `www.company.com:80` is a superset of `www.company.com:80/sales`, Oracle Web Cache matches requests for `www.company.com:80/sales` to site definition `www.company.com:80` rather than `www.company.com:80/sales`. In addition, Oracle Web Cache uses the site-to-server mapping for `www.company.com:80`.

To avoid this problem, you would have to configure the site definitions in the following order:

```
www.company.com:80/sales
www.company.com:80
```

After you create site definitions, create ordered mappings of sites to origin servers. To avoid requests being mapped to the wrong site, you must be careful in how you order these mappings:

- 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` do not resolve as expected. Requests for these URLs 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
  ```

For instructions on creating site definitions and site-to-server mappings, see Section 2.11.3 and Section 2.11.4.

2.3 About Resource Limits in Oracle Web Cache Management

As a part of configuration, specify caching and network thresholds to ensure Oracle Web Cache runs efficiently.

Oracle Web Cache provides the following types of configurable thresholds:

- Section 2.3.1, "Maximum Cache Size"
- Section 2.3.2, "Maximum Incoming Connections"
- Section 2.3.3, "Maximum Cached Object Size"
- Section 2.3.4, "Network Timeouts"

2.3.1 Maximum Cache Size

When the maximum cache memory limit is reached, Oracle Web Cache performs **garbage collection**. During garbage collection, Oracle Web Cache removes stale objects based on **popularity** and **validity**. In a cache cluster environment, Oracle Web Cache removes on-demand objects before it removes owned objects.

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 Oracle Web Cache should be set to at least 512 MB.

Your application's memory requirements vary based upon factors, such as object size, number of objects, the number of HTTP headers returned, and whether **ESI** is present. To get a close approximation on the maximum amount of memory required, you may apply the formula provided below.

Most customers leave this setting to the default which is 500 MB. If want to change the default, perform the following steps to determine the maximum amount of memory required:

1. Use the following formula to determine an estimate of the maximum memory, in bytes, needed

   ```
   1.25*(TotalDocs * ((AvgDocSize/8192+1) *8192 + 16384))
   ```

   In the formula:

   - `.25` accounts for the run time memory overhead.
   - *`TotalDocs`* is the total number of objects you intend to store in the cache.
   - *`AvgDocSize`* is the average size of objects, in bytes, you intend to store in the cache. You can determine the average size by viewing the following metrics on the Performance Summary page.
     - **Performance of each Site with Summary** > ***site*** > **Cache Size**
     - **Performance of each Site with Summary** > ***site*** > **Number of Cached**

     See Section 8.4 for further information about the Performance Summary page.

   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.

2. Convert the result to megabytes.
3. Specify the estimated memory in the Oracle Web Cache configuration. See Section 2.11.5.
4. 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 Oracle 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 Performance Summary page of Fusion Middleware Control provides information about the current memory use and the maximum memory use. To access this page:

   a. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
   b. From the **Web Cache** menu, select **Monitoring** and then **Performance Summary**.

2.3.2 Maximum Incoming Connections

In addition to the cache size, it is important to specify a reasonable number for the maximum connection limit for the Oracle Web Cache server. The default is 500. 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, Oracle Web Cache serves fewer

fewer concurrent requests. 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` or LoadRunner, 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. See Section 3.6.3 to configure this setting in Fusion Middleware Control and Section 3.7.1 to configure this setting in Oracle Web Cache Manager.

Use various tools, such as those available with the operating system and with Oracle 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. The Web Cache Home page and the Performance Summary page in Fusion Middleware Control provide statistics on hits and misses. From **Web Cache** menu, select **Home** and **Monitoring** > **Performance Summary** to access these page.

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

To specify the maximum number of incoming connections, see Section 2.11.5.

Connections on UNIX

On most UNIX platforms, each client connection requires a separate file descriptor. Oracle Web Cache tries to reserve the maximum number of file descriptors (`Max_File_Desc`) when it starts. If the Oracle 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, Oracle Web Cache fails to start.

On most UNIX platforms, each client connection requires a separate file descriptor. The Oracle Web Cache server attempts to reserve the maximum number of file descriptors when it starts. If you have root privileges, you can increase this number. For example, for the LINUX Red Hat Operating System you can increase the maximum number of file descriptors by modifying Oracle Web Cache users file descriptors limits in `/etc/security/limits.conf`.

For example to allow the user `WC_USER` to have 4092 connections, in the `/etc/security/limits.conf` file add the following entries:

```
WC_User       soft    nofile          4092
WC_User       hard    nofile          4092
```

Make sure the parameter `fs.file-max` is set to 65k in the `/etc/sysctl.conf`. On Solaris Operating System you can increase the maximum number of file descriptors by setting the `rlim_fd_max` parameter. If `webcached` is not run as root, the Oracle Web Cache server logs an error message and fails to start.

For instructions on changing the `webcached` executable to run with the root privilege, see Section 5.9.

For more information, see:

- Operating system-specific documentation for connection limitations
- *Oracle Fusion Middleware Performance Guide* for TCP/IP performance tuning tips

Connections on Windows

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

The default maximum number of TCP ports is set to 5000 by the operating system. Of those, 1024 are reserved by the kernel. You can modify the maximum number of ports by editing the Windows registry. Windows operating systems allow up to 65534 ports.

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

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

Add a new value, specifying the following:

- Value Name: `MaxUserPort`
- Value Type: `DWORD`
- Value Data: 65534
- Valid Range: 5000 to 65534

On Windows operating systems, Oracle Web Cache does not attempt to reserve file handles or to check that the number of current maximum incoming connections is less than the number of TCP ports.

2.3.3 Maximum Cached Object Size

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, the cache only stores objects that are not larger than a specified size and that match the caching rules. Oracle Web Cache does not cache objects larger than the specified size, even if they match caching rules. The default is 100 KB. 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, Oracle Web Cache does not cache any objects, effectively turning off caching.

To specify the size of single cached object, see Section 2.11.5.

2.3.4 Network Timeouts

Oracle Web Cache enables you to specify settings for the following timeouts:

- Keep-Alive Timeout: The keep-alive timeout is the time limit for the client to process a request from Oracle Web Cache. After Oracle 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 Oracle Web Cache. If the network between the client and Oracle Web Cache is slow, consider increasing the keep-alive timeout.
- Client Send: Specifies the allowed time for Oracle Web Cache to finish a send operation to the client.
- Client Receive: Specifies the allowed for Oracle Web Cache to wait for a receive operation to complete from the client.
- Origin Server Send: Specifies the time allowed for the Oracle Web Cache to finish a send operation to the origin server.
- Origin Server Receive: Specifies the time allowed for the origin server to generate and start sending a response to Oracle Web Cache.
- Origin Server Connect: Specifies the time allowed for Oracle Web Cache to complete connection establishment to an origin server. If an origin server has multiple IP addresses (for example., IPv4 and IPv6) that will be retried, the timeout refers to connecting to one origin server IP address. If the origin server cannot generate a response within that time, Oracle Web Cache drops the connection and sends a network error page to the client. If applications require a shorter timeout, adjust the timeout.

To specify the timeouts, see Section 2.11.5.

2.4 About Oracle Web Cache Ports

Ports are dynamically assigned to many components when they are provisioned. The port numbers stay the same after the provisioning unless they are manually changed.

Oracle Web Cache uses a HTTP or a HTTPS listening port to receive requests. In addition to a listening port, Oracle Web Cache also receives administration, invalidation, and statistics monitoring requests on specific HTTP or HTTPS listening ports.

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

To configure port settings, see Section 2.11.1.

2.5 About IP Addresses

Oracle Web Cache supports both IP version 4 and version 6 addresses.

The following examples show IP version 4 addresses:

- 138.1.16.102 specifies an IP address.
- 138.1.16.102/255.255.0.0 specifies an IP address and subnet mask, 138.1.*any*.*any*. The zeros in the mask mean that any value is OK. This address is equivalent to 138.1.0.0/255.255.0.0 or 138.1.*.*.
- 138.1.16.102/16 is another way to specify the previous example. It means that only the high 16 bits matter.

You must use the wildcard for an entire field. Therefore, you cannot use a wildcard to specify something like 138.128.0.0/255.128.0.0. In this example, the high 9 bits need to

be checked. 138.128*.*.* is not allowed. 138.*.*.* would check only the high 8 bits, and the other 3 8-bit fields could have any value.

The following examples show IP version 6 addresses:

- FE80:0:0:0:205:2FF:FE71:2594 specifies an IP address.
- FE80:0:0:0:205:0:0:0/FFFF:FFFF:FFFF:FFFF:FFFF:0:0:0 specifies an IP address and subnet mask. Here the high 16*5 = 80 bits, where 16*5 is 16 bits from each FFFF * 5 fields of FFFF, need to be checked for a match. If you prefer to use a wildcard, you can specify the same address as FE80:0:0:0:205:*:*:*.
- FE80:0:0:0:205:0:0:0/80 is another way to specify the previous example. It also specifies to check the high 80 bits.

You must use the wildcard for an entire field.

2.6 Getting Started with Managing Oracle Web Cache with Oracle Enterprise Manager Fusion Middleware Control

Oracle Enterprise Manager Fusion Middleware Control enables you to manage Oracle Fusion Middleware components in a farm, including Oracle Web Cache.

A **farm** is a collection of components managed by Fusion Middleware Control. It can contain Oracle WebLogic Server domains, one Administration Server, one or more Managed Servers, and the Oracle Fusion Middleware components that are installed, configured, and running in the domain.

From Fusion Middleware Control, you can configure the following:

- Properties for the cache, including origin servers, sites, resource limits, passwords, sessions, and cookies
- Event logging and access logging
- Multiple caches into one **cache cluster** for shared configuration, scalability, and high availability
- Request filtering to block invalid or illegal HTTP requests
- Caching rules and expiration policies

Oracle Web Cache Manager provides additional configuration areas. See Section 2.7 for more information about using Oracle Web Cache Manager.

You can also monitor performance statistics and perform operational tasks, such as starting and stopping a cache, synchronizing configuration among cache cluster members using Fusion Middleware Control.

This section covers the following topics:

- Section 2.6.1, "Logging into Fusion Middleware Control"
- Section 2.6.3, "Understanding Statistics on the Web Cache Home Page"
- Section 2.6.4, "Using the Fusion Middleware Control Help"

For general information about Fusion Middleware Control, see the *Oracle Fusion Middleware Administrator's Guide*.

2.6.1 Logging into Fusion Middleware Control

To view the Oracle Web Cache pages:

1. To display Fusion Middleware Control, you enter the Fusion Middleware Control URL, which includes the name of the host and the port number assigned to Fusion Middleware Control during the installation. The following shows the format of the URL

   `http://hostname.domain:port/em`

   The port number is the number of the Administration Server of Oracle WebLogic Server. By default, the port number is 7001.

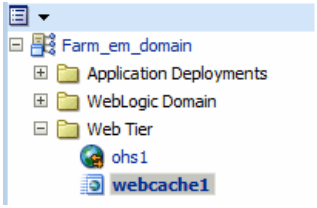
2. Enter the Oracle Fusion Middleware administrator user name and password and click **Login**.

   The default user name for the administrator user is `weblogic`. The password is the one you supplied during the installation of Oracle Fusion Middleware.

2.6.2 Navigating to Oracle Web Cache Administration Pages

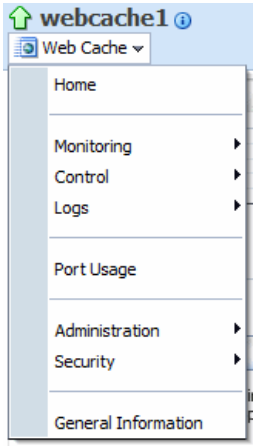
To navigate to Oracle Web Cache administration tasks:

1. From the navigation pane, expand the farm and then the **Web Tier** installation type, and select Oracle Web Cache component.



   The Web Cache home page displays. See Section 2.6.3 for further information about the contents of the home page.

2. Select the **Web Cache** menu.



   The Web Cache menu displays the following options described in Table 2–1

Table 2–1 Web Cache Menu Options

Element	Description
Home	This option displays the Web Cache Home page. See Section 2.6.3 for further information about the contents of this page.

Table 2–1 (Cont.) Web Cache Menu Options

Element	Description
Monitoring	This option displays the following options: ■ **Performance Summary:** This option enables you to view performance metrics for Oracle Web Cache. See Section 8.4 for further information. ■ **Popular Requests:** This option enables you to view the most popular requests for determining if the caching rules are caching the correct objects. See Section 8.6 for further information.
Control	This option provides options for starting, stopping, and restarting Oracle Web Cache. See Section 2.13.2 for further information.
Logs	The **View Log Message** option displays the Log Messages page for viewing the contents of event log files. See Section 9.7 for further information.
Port Usage	This option display the ports in use. See Section 2.11.1 for more information.

Table 2–1 (Cont.) Web Cache Menu Options

Element	Description
Administration	This option displays the following options: ■ **Caching Rules:** This option enables you to configure caching rules. See Chapter 6, "Caching and Compressing Content," for more information. ■ **Request Filters:** This option enables you to configure request filters for protecting against common HTTP request attacks. See Chapter 4, "Configuring Request Filtering," for more information. ■ **Expiration:** This option enables you to specify expiration policies. See Section 6.7 for more information. ■ **Sites:** This option enables you to configure the Web sites for which Oracle Web Cache caches content. See Section 2.11.3 for more information. ■ **Origin Servers:** This option enables you to configure to configure application Web servers or proxy servers to which Oracle Web Cache sends cache misses. See Section 2.11.4 for more information. ■ **Cluster:** This option enables you to configure a **cache cluster**. See Section 3.6 for more information. ■ **Passwords:** This option enables you to configure security options for Oracle Web Cache. See Chapter 5, "Configuring Security," for more information. ■ **Access Logs:** This option enables you to configure settings for Oracle Web Cache access logs. See Chapter 9, "Logging," for more information. ■ **Event Logs:** This option enables you to configure settings for Oracle Web Cache event logs. See Chapter 9, "Logging," for more information. ■ **Resource Limits:** This option enables you to configure cache thresholds for Oracle Web Cache. See Section 2.11.5 for more information. ■ **Passwords:** This option enables you to configure security options for Oracle Web Cache. See Chapter 5, "Configuring Security," for more information. ■ **Session Configuration:** This option enables you to create session definitions for: - Specifying how session requests are served by the cache - Enabling **session binding**, whereby a user session for a particular site is bound to an origin server to maintain state for a period - Substituting session information in **session-encoded URLs** See Section 2.12 for more information. ■ **Multi-Version Cookies:** This option enables you to specify cookie values for multiple-version URLs. See Section 6.5.1 for more information. ■ **Ports Configuration:** This option enables you to configure ports for Oracle Web Cache. See Section 2.11.1 for more information.

Table 2–1 (Cont.) Web Cache Menu Options

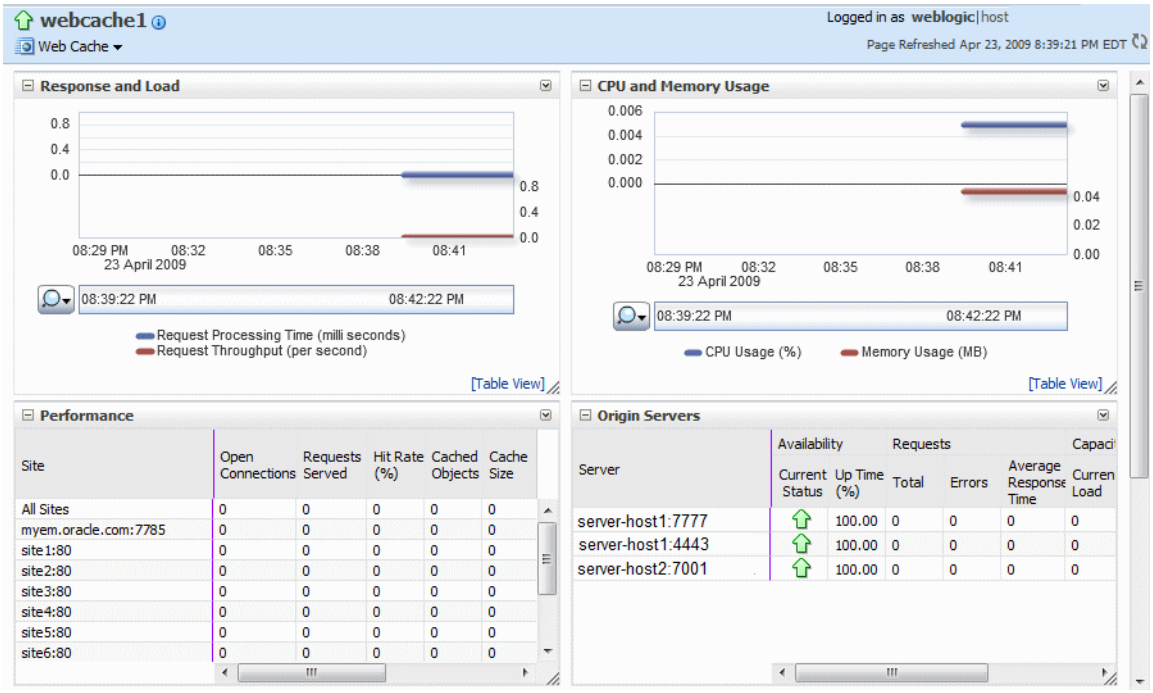
Element	Description
Security	This option displays the following options: ▪ **Audit Policy:** This option enables you to configure audit policy settings. See the *Oracle Fusion Middleware Security Guide*. ▪ **SSL Configuration:** This option enables you to SSL configuration for Oracle Web Cache. See Section 5.4.2 and Section 5.4.3 for more information. ▪ **Wallets:** This option enables you to create and manage wallets. See Section 5.4.1 for more information.
General Information	This option displays general details about the instance.

2.6.3 Understanding Statistics on the Web Cache Home Page

The Web Cache Home page provides general information about the selected cache system component, as well as the supported sites and origin server.

You can also use this page as a starting point for monitoring and administering Oracle Web Cache. Figure 2–1 shows a portion of the Web Cache Home page.

Figure 2–1 Web Cache Home Page



This page contains the following statistical regions:

- Response and Load Region
- CPU and Memory Usage Region
- Performance Region
- Origin Servers Region

Response and Load Region

Table 2–2 describes the performance-monitoring statistics in the **Response and Load** region.

Table 2–2 Response and Load Statistics

Statistic	Description
Request Processing Time	This metric specifies the average number of milliseconds used to process requests.
Request Throughput	This metric specifies the average number of requests served for each second.

CPU and Memory Usage Region

Table 2–3 describes the performance-monitoring statistics in the **Response and Load** region.

Table 2–3 CPU and Memory Usage Statistics

Statistic	Description
CPU Usage	This metric specifies the percentage of the CPU that is being used for Oracle Web Cache. As traffic increases, CPU utilization increases.
Memory Usage	This metric specifies the total memory used by the Oracle Web Cache component.

Performance Region

Table 2–4 describes the performance-monitoring statistics in the **Origin Servers** region.

Table 2–4 Performance Statistics

Statistic	Description
Open Connections	This metric specifies the current number of incoming open connections to the Oracle Web Cache server.
Requests Served	This metric specifies the accumulated number of requests that Oracle Web Cache has served since it was started.
Hit Rate	This metric specifies the percentage of requests resolved by cache content.
Cached Objects	This metric specifies the total number of objects stored in the cache.
Cache Size	This metric specifies the size, in megabytes, of the objects currently stored in the cache. For a cache cluster member, this number is an aggregate of the owned and on-demand objects.
Requests Denied by Request Filtering	This metric specifies the accumulated number of requests denied by request filters. Any nonzero number may be an indication of an attack on the site or an issue with the configuration of request filters.
Bytes Saved by Compression	This metric specifies the accumulated number of bytes that would be sent to clients if in-cache compression is disabled.
Error Pages Served	This metric specifies the accumulated number of error pages that Oracle Web Cache served to Web browsers since the cache was started.

Origin Servers Region

Table 2–5 describes the performance-monitoring statistics in the **Origin Servers** region.

Table 2–5 Origin Server Statistics

Statistic	Description
Server	This metric displays the name of the origin server.
Current Status	This metric displays the status of the origin server when Oracle Web Cache last attempted to communicate with that origin server. (Oracle Web Cache attempts to reach the origin server only for specific purposes, such as retrieving responses for a cache miss.)
Up Time (%)	This metric is the up time is from the perspective of Oracle Web Cache. It is an approximation of the origin server's uptime. The accuracy is based on how long Web Cache has been up and how often Oracle Web Cache sends requests to that origin server.
Requests	This metric category provides the following metrics: ■ **Total:** This metric specifies the accumulated number of requests that the origin server has processed. ■ **Errors:** This metric specifies the number of errors encountered from this origin server by Oracle Web Cache. These errors may include: - Oracle Web Cache failed to connect to the origin server. - There was an error transmitting the request or receiving the response to or from the origin server - There was an HTTP 500 class response from the origin server. ■ Average Response Time The metric specifies the average time the origin server has taken to process and reply to requests which it has received from this Oracle Web Cache.
Capacity	This metric category provides the following metrics: ■ **Current Load:** The metric specifies the current number of connections from Oracle Web Cache that the origin server has open. ■ **Maximum Load:** The metric specifies the maximum number of connections that the origin server has had open simultaneously. ■ **Configured:** The metric specifies the capacity set for the server in the Origin Servers page. **Note:** If the value for the **Maximum Load** metric is close to the **Configured** metric, then increase the capacity in the Origin Servers page. See Section 2.11.2.

2.6.4 Using the Fusion Middleware Control Help

The **Oracle Enterprise Manager Help** command on the **Help** menu provides users with access to task-related or conceptual information relating to the current Fusion Middleware Control page. In addition, you can click a **Help** icon on some pages, where further explanation of page elements is necessary.

There is also a search feature, allowing you to search the help and selected Oracle Fusion Middleware documents that are included with the online help system. The help guides you to specific, context-sensitive information in these documents.

2.7 Getting Started with Managing Oracle Web Cache with Oracle Web Cache Manager

Oracle Web Cache Manager is a graphical user interface tool that provides configuration capabilities for the following areas not provided by Fusion Middleware Control:

- Invalidation
- Advanced options for request filters
- Network thresholds for Oracle Web Cache
- Advanced security options
- Invalidation
- Diagnostics features
- Error pages to be served by Oracle Web Cache

This section introduces you to the features of Oracle Web Cache Manager. This section contains these topics:

- Section 2.7.1, "Starting Oracle Web Cache Manager"
- Section 2.7.2, "Navigating Oracle Web Cache Manager"

2.7.1 Starting Oracle Web Cache Manager

To start Oracle Web Cache Manager:

1. Configure a secure password for the Oracle Web Cache administrator with the monitor account. You use the password for the monitor account to log in to Oracle Web Cache Manager. See Section 5.2 to set a secure password.
2. Start the **admin server process** with the following command:

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

   `WebCache-admin` represents the `admin` server process.
3. Determine the port for the `admin` server process. See Section 2.11.1.
4. Point your browser to the following URL:

   ```
   http://web_cache_hostname:admin_port/webcacheadmin
   ```

   See Section 2.11.1.2 to determine the port.
5. Enter the Oracle Web Cache administrator user name, `administrator` and the password you set in Step 1.

2.7.2 Navigating Oracle Web Cache Manager

The Oracle Web Cache Manager interface includes:

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

Figure 2–2 shows the Oracle Web Cache Manager interface.

Figure 2–2 Oracle Web Cache Manager Interface

The interface contains the following features:

- The **Home** link directs you to a welcome page.
- **Navigation Pane** provides a graphical tree view of configuration, administration, and performance monitoring capabilities for Oracle Web Cache and its supported Web sites.
- **Content Pane** shows the property sheet for the selected option from the navigation pane.
- The **Apply Changes** button applies submitted static and dynamic configuration changes to Oracle Web Cache; the **Cancel Changes** button cancels submitted static and dynamic configuration changes to Oracle Web Cache.
- **Status Message** provide the following possible status messages:
  - **Web Cache running with current configuration:** This message appears if Oracle Web Cache is running with an up-to-date configuration.
  - **Web Cache running in Routing Only mode with current configuration:** This message appears if Oracle Web Cache is running with an up-to-date configuration.
  - **Press "Apply Changes" to commit your modifications:** This message appears if Submit has been selected in some dialog box, but the Apply Changes button has not been chosen.
  - **Restart Web Cache to make configuration changes take effect:** This message appears if Oracle Web Cache is running with an older version of the configuration. This can happen when static configuration changes have been applied to `webcache.xml`, but Oracle Web Cache was not restarted.

- **Dynamic Changes Applied. Restart Not Needed:** This message appears if one or more dynamic configuration changes were applied, which do not require a restart of Oracle Web Cache.
- **Retrieve configuration from remote cache:** This message appears if the cache has been recently upgraded to the current version of Oracle Web Cache but the configuration has not been copied to the local cache configuration file.

The navigator frame contains the following major categories described in Table 2–1. Additional categories and options are available, but Fusion Middleware Control provides the preferred functionality in these areas.

Table 2–6 Web Cache Manager Navigation Pane Options

Category	Description
Operations	This category contains the following options: ■ **Cache Operations:** This option displays the Cache Operations page for starting, stopping, or restarting the `cache` server process. See Section 2.7.3 for further information. ■ **Basic Content Invalidation:** and **Advanced Content Invalidation:** These options enable you to invalidate content in the cache. See Section 7.7.2.1 for further information. ■ **On-Demand Log File Rollover:** This option enables you to immediately roll over event and access logs. See Section 9.8 for further information.
Filtering	The preferred method for configuring request filters is using Fusion Middleware Control, as described in Chapter 4, "Configuring Request Filtering." Use the **Request Filters** option in Oracle Web Cache Manager to copy rules and revert configuration settings, as described in Section 4.14.
Properties	This category contains the following options: ■ **Security:** This option provides advanced security options. See Chapter 5 for further information. ■ **Network Timeouts:** This option enables you to configure the network setting for connections. See Section 2.11.5 for further information.
Logging and Diagnostics	This category contains the following option: ■ **Diagnostics:** This option enables diagnostics information to display in the HTML response body of an object. See Section 8.8 for further information.

2.7.3 Understanding the Cache Operations Page

The Cache Operations page of Oracle 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.

2.8 Getting Started with Managing Oracle Web Cache with Oracle Process Manager and Notification (OPMN)

Oracle Process Manager and Notification (OPMN) Server manages Oracle Web Cache processes, including the **admin server process** and **cache server process**:

- The `admin` server process transfers the contents of the `webcache.xml` configuration file between the Oracle Web Cache instance and the Oracle WebLogic Server environment where Fusion Middleware Control is running.
- The `cache` server process manages the cache.

OPMN provides the `opmnctl` command. The command is located in the following directory:

```
(UNIX) ORACLE_INSTANCE/bin/
(Windows) ORACLE_INSTANCE\bin
```

To get started with OPMN, use the `opmnctl` command to query the status of the components in your installation and obtain a list of all the ports in use:

```
opmnctl status -l
```

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

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

Table 2–7 shows the commands of the `opmnctl` utility that are applicable to Oracle Web Cache.

Table 2–7 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 2–8.
`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 2–8 shows the parameters for the `opmnctl` utility. It also shows the valid values that are applicable for Oracle Web Cache. Unless otherwise noted, you can use any parameter with any command, except for `status`, listed in Table 2–7.

Table 2–8 Parameters for the opmnctl Utility

Parameter	Valid Values	Description
`ias-component=`*`component_name`*	Oracle Web Cache instance name	Takes the specified action for the Oracle Web Cache `admin` server process and `cache` server process. For example, the following command starts both the Oracle Web Cache `admin` server and `cache` server processes on system component `webache1`: `opmnctl startproc ias-component=webcache1` You must always specify this parameter to administer any Oracle Web Cache process.
`process-type=`*`value`*	`WebCache` `WebCache-admin`	Takes the specified action for the process specified in the value: ▪ `WebCache`: The `cache` server process ▪ `WebCache-admin`: The `admin` server process The parameter `ias-component=`*`component_name`* must precede this parameter. For example, the following command starts only the `cache` server process for Oracle Web Cache `webcache1`: `opmnctl startproc ias-component=webcache1 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, Oracle 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, Oracle 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=`*`component_name`* must precede this parameter.

For additional information about using OPMN and its supported commands, see *Oracle Fusion Middleware Oracle Process Manager and Notification Server Administrator's Guide*.

2.9 Basic Tasks for Configuring and Managing Oracle Web Cache

When you configure an environment with Oracle Web Cache, you first ensure the Oracle Web Cache component is added to the installation. If it is not, add the Oracle Web Cache component to the configuration.

The following provides a summary of the steps to configure and manage a basic Oracle Web Cache:

1. Add Oracle Web Cache system component to an environment. See Section 2.10.

   To create a cache cluster, see Section 3.6.

2. Specify properties for an Oracle Web Cache instance after it is added to an installation. See Section 2.11.

3. Configure secure passwords for Oracle Web Cache. See Section 5.2

4. Configure session definitions for session-related properties. See Section 2.12.

5. Configure access and event logs. See Chapter 9, "Logging."

6. Configure request filtering. See Chapter 4, "Configuring Request Filtering."
7. Configure and monitor caching rules. See Section 6.6.
8. Invalidate content. See Chapter 7, "Invalidating Content."

2.10 Adding an Oracle Web Cache System Component to an Environment

For an Oracle Web Tier or an Oracle Portal, Forms, Reports and Discoverer installation in which Oracle Web Cache was not selected, you can easily add an Oracle Web Cache system component, because the Oracle Universal Installer installs the necessary software.

To add an Oracle Web Cache system component to an installation:

1. From the command line, go the following directory:

   ```
   (UNIX) ORACLE_INSTANCE/bin
   (Windows) ORACLE_INSTANCE\bin
   ```

2. Create the Oracle Web Cache system component:

   ```
   createcomponent -componentType WebCache -oracleHome ORACLE_HOME -oracleInstance
   ORACLE_INSTANCE -componentName component_name
   ```

   For example to create an Oracle Web Cache system component named `webcache2`, you would enter syntax similar to the following:

   ```
   createcomponent -componentType WebCache -oracleHome /scratch/webtier
   -oracleInstance /scratch/instances/instance1 -componentName webcache2
   ```

3. Configure caching rules. See Section 6.6.
4. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
5. View the **Origin Servers** section to ensure requests for the origin server are going through the Oracle Web Cache.

2.11 Specifying Properties for an Oracle Web Cache System Component

To establish properties for an Oracle Web Cache system component, perform the following tasks:

- Section 2.11.1, "Task 1: Configure Port Configuration for Oracle Web Cache"
- Section 2.11.2, "Task 2: Specify Origin Server Settings"
- Section 2.11.3, "Task 3: Specify Site Definitions"
- Section 2.11.4, "Task 4: Map Site Definitions to Origin Servers"
- Section 2.11.5, "Task 5: Set Resource Limits and Network Thresholds"
- Section 2.11.6, "Task 6: Configure Error Pages"
- Section 2.11.7, "Task 7: Restart Oracle Web Cache"

2.11.1 Task 1: Configure Port Configuration for Oracle Web Cache

Oracle Web Cache uses a HTTP or HTTPS listening port to received requests. You can add listening ports, if necessary. For example, it may be necessary to add a listening

port to assign Oracle Web Cache a port that an origin server was previously listening on.

In addition to a listening port, Oracle Web Cache also receives requests for the **admin server process**, invalidation, and statistics monitoring requests on specific HTTP or HTTPS listening ports. You can modify these operation ports.

This section contains the following topics related to port configuration for Oracle Web Cache:

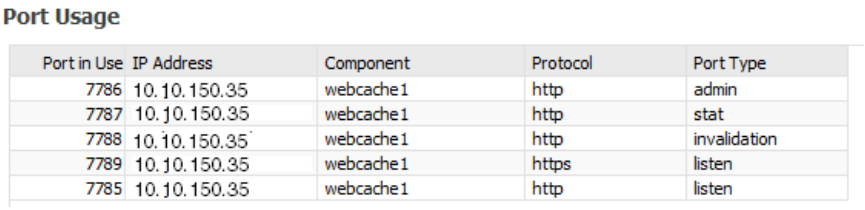
- Section 2.11.1.1, "Verifying Port Configuration for Oracle Web Cache with Fusion Middleware Control"
- Section 2.11.1.2, "Verifying Port Configuration for Oracle Web Cache with OPMN"
- Section 2.11.1.3, "Adding an Oracle Web Cache Listening Port"
- Section 2.11.1.4, "Modifying Oracle Web Cache Operation Ports"

2.11.1.1 Verifying Port Configuration for Oracle Web Cache with Fusion Middleware Control

To determine ports in use by Oracle Web Cache with Fusion Middleware Control:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** > **Port Usage**.

   The Ports Usage page displays.



Port Usage

Port in Use	IP Address	Component	Protocol	Port Type
7786	10.10.150.35	webcache1	http	admin
7787	10.10.150.35	webcache1	http	stat
7788	10.10.150.35	webcache1	http	invalidation
7789	10.10.150.35	webcache1	https	listen
7785	10.10.150.35	webcache1	http	listen

   In this example:

   - Port 7785 is the HTTPS listening port for Oracle Web Cache.
   - Port 7786 is the HTTP listening port for the `admin` server process.
   - Port 7787 is the HTTP listening port for statistics monitoring requests.
   - Port 7788 is the HTTP listening port for invalidation requests.
   - Port 7789 is the HTTP listening port for Oracle Web Cache.

2.11.1.2 Verifying Port Configuration for Oracle Web Cache with OPMN

To determine ports in use by Oracle Web Cache with OPMN:

```
(UNIX) ORACLE_INSTANCE/bin/opmnctl status -l
(Windows) ORACLE_INSTANCE\bin\opmnctl status -l
Processes in Instance: instance1
---------------------------------+--------------------+---------+----------+------
------+----------+-----------+------
ias-component                    | process-type       |     pid | status   |
uid |  memused |    uptime | ports
---------------------------------+--------------------+---------+----------+------
------+----------+-----------+------
webcache1                        | WebCache-admin     |   11244 | Alive    |
```

```
458185369 |    45852 | 180:27:00 | http_admin:7786
webcache1                        | WebCache           |   11243 | Alive    |
458185368 |    72540 | 180:27:00 | http_stat:7787,http_invalidation:7788,https_
listen:7789,http_listen:7785
ohs1                             | OHS                |    6077 | Alive    |
458185358 |   349220 | 180:48:14 | https:9999,https:4443,http:7777
```

In this example:

- Port 7785 is the HTTP listening port for Oracle Web Cache, represented by `http_listen`.
- Port 7786 is the HTTP listening port for the `admin` server process, represented by `WebCache-admin`
- Port 7787 is the HTTP listening port for statistics monitoring requests, represented by `http_stat`.
- Port 7788 is the HTTP listening port for invalidation requests, represented by `http_invalidation`.
- Port 7789 is the HTTP listening port for Oracle Web Cache, represented by `https_listen`.

2.11.1.3 Adding an Oracle Web Cache Listening Port

You can add listening ports, if necessary. For example, it may be necessary to add listening port to assign Oracle Web Cache a port that an origin server was previously listening on. If want to configure an HTTPS port, see Section 5.4.2.

To add an HTTP listening port:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** > **Ports Configuration.**

   The Ports Configuration page displays.
3. Click **Create**.

   The Create Port page appears.
4. From the **Port Type** list, select **NORM**.
5. In the **IP Address** field, specify the computer running Oracle Web Cache:
   - IP version 4 address written in a 32-bit dotted decimal notation or an IP version 6 address written in a 128-bit notation. See Section 2.5.
   - A host name that resolves to an IP address of the computer running Oracle 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** field, enter the listening port from which Oracle Web Cache receives 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. To configure Oracle Web Cache to listen on a port less than 1024, such as on port 80, run the Oracle Web Cache `webcached` executable with the root privilege. If the `webcached` executable is not run as root, Oracle Web Cache fails to start.

See Section 5.9 for instructions on changing the `webcached` executable to run as root.

7. If you are changing the listening port from an HTTP port to an HTTPS port, see Section 5.4.2 to configure SSL settings.
8. Click **OK**.

2.11.1.4 Modifying Oracle Web Cache Operation Ports

To modify ports from which Oracle Web Cache receives administration, invalidation, or statistics monitoring requests:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** > **Ports Configuration.**

   The Ports Configuration page displays.
3. Select the port you want to modify and click **Edit**.

   The Edit Port page appears.
4. In the **Endpoint Attributes** section, from the **Port Type** list, select **ADMINISTRATION**, **INVALIDATION**, or **STATISTICS**.
5. In the **IP Address** field, specify the computer running Oracle Web Cache:
   - IP version 4 address written in a 32-bit dotted decimal notation or an IP version 6 address written in a 128-bit notation. See Section 2.5.
   - A host name that resolves to an IP address of the computer running Oracle 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** field, enter the listening port from which Oracle Web Cache receives 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. To configure Oracle Web Cache to listen on a port less than 1024, such as on port 80, run the Oracle Web Cache `webcached` executable with the root privilege. If the `webcached` executable is not run as root, Oracle Web Cache fails to start.

   See Section 5.9 for instructions on changing the `webcached` executable to run as root.
7. If you are changing a port from an HTTP port to an HTTPS port, see Section 5.5.1 to configure SSL settings.
8. Click **OK**.

2.11.2 Task 2: Specify Origin Server Settings

Configure Oracle Web Cache with the application Web servers or proxy servers to which it sends cache misses. Typically, Oracle Web Cache uses application Web servers for internal sites and proxy servers for external sites outside a firewall.

If Oracle HTTP Server was installed, the installation process creates a default origin server based on the host name and listening port of Oracle HTTP Server.

Oracle Web Cache only forwards requests to a configured origin server if the origin server is mapped to a Web site.

When you configure multiple origin servers, ensure the host and port settings are not identical. If you configure origin servers with duplicate host and port settings, both the `cache` server and `admin` server processes fail to start.

To configure Oracle Web Cache with origin server information:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Origin Servers**.

   The Origin Servers page displays.
3. Click **Create**.

   The Create Origin Server page displays.
4. Configure the **Host**, **Port**, **Capacity**, **Protocol**, and **Routing Enabled** settings for each origin server to send Oracle Web Cache requests using the descriptions in Table 2–9.
5. For configurations with multiple origin servers, specify how you want to HTTP distribute requests Oracle Web Cache sent to other origin servers when there is a failure in the **Failover** section using the descriptions in Table 2–9.
6. Specify these settings if these origin server is a proxy server in the **Proxy Web Server** section using the descriptions in Table 2–9.
7. Click **OK** to apply changes and return to the Origin Servers page. It is not necessary to click **Apply** in the Origin Servers page to apply this change.

Table 2–9 Create Origin Server

Element	Description
Host	Enter the host name of the origin server.
Port	Enter the listening port from which the origin server receives Oracle Web Cache requests. **Note:** Oracle 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 proxied port.
Capacity	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 back-end database reaches full capacity. In a cache cluster, Oracle 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`
Protocol	Select either **HTTP** to send HTTP requests on the port or **HTTPS** to send HTTPS requests on the port.

Table 2–9 (Cont.) Create Origin Server

Element	Description
Routing Enabled	Click to permit Oracle Web Cache to route requests to the origin server or leave unchecked to only serve requests from cache. Oracle recommends not selecting this option if temporary maintenance of an origin server is needed. Oracle 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 Routing Enabled not selected, Oracle Web Cache serves a network error page to clients. See Section 2.11.6 for further information about configuring error pages.
Failover Threshold	Enter the number of allowed continuous read and write failures with an origin server on established connections. The default is five request and response failures. If any connection failure occurs, Oracle Web Cache immediately considers an origin server down. When the threshold is met, Oracle Web Cache considers the origin server down and performs automatic failover of the origin servers. If an origin server fails at any time after Oracle Web Cache has started to send a request, then Oracle Web Cache increments the failure counter. The failure counter is reset if there is 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 something other than 1xx, 2xx, 3xx, 4xx, 501 Not Implemented, and 505 HTTP Version Not Supported. After the threshold is met, Oracle Web Cache considers the server down and uses other servers for future requests. Oracle Web Cache starts polling the down server, by sending requests to the URL specified in the **Ping URL** field. When Oracle 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, Oracle Web Cache considers the server up again and uses it for future requests. **Notes:** ■ The threshold does not apply if Oracle Web Cache cannot connect to an origin server. In this case, Oracle Web Cache immediately considers the server down and does not use it for future requests. If there are other origin servers, Oracle Web Cache retries the request to another origin server. If there no servers configured, Oracle Web Cache returns an error. ■ The failover to another origin server does not apply if there is only one origin server left.

Table 2–9 (Cont.) Create Origin Server

Element	Description
Ping URL	Enter the URL that Oracle Web Cache uses 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. The default value is: / 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.
Ping Frequency (seconds)	Enter the time, in seconds, that Oracle Web Cache uses to poll an origin server that has reached its failover threshold. The default is 10 seconds.
Proxy Web Server	Click to treat this origin server as a proxy server.
Username	Enter the user name for the proxy server administrator.
Password	Enter the password of the proxy server administrator
Confirm Password	Reenter the password for the proxy server administrator.

2.11.3 Task 3: Specify Site Definitions

For Oracle Web Cache to act as a virtual server for one or more Web sites, configure Oracle Web Cache with information about the named Web sites. For an overview of site configuration, see Section 2.2.

To create site definitions:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Sites**.

   The Sites page displays.
3. From the **Site Definitions** section, click **Create**.

   The Create Site page displays.
4. In the Create section, configure the elements using the descriptions in Table 2–10.
5. In the **Aliases** section, specify all the possible aliases for the site to ensure requests are directed to the correct site. An alias specifies the host and port in which browsers use to connect to the site.
   a. Click **Create** to create an alias.
   b. Configure the **Host** and **Port** fields using the descriptions in Table 2–10.
6. Click **OK** to apply changes and return to the Sites page. It is not necessary to click **Apply** in the Sites page to apply this change.
7. Repeat Steps 3 to 6 for each additional site.

8. In the Sites page, use the **Move Up** and **Move Down** icons to order the definitions.

   Oracle Web Cache resolves an incoming request first to a site definition, and then to the first matching site-to-origin server mapping. See Section 2.2 for more information about how Oracle Web Cache uses the order of site definitions and site-to-server mappings to match requests.

9. Click **Apply** to apply the move change.

Table 2–10 Create Site Page

Element	Description
Host	In the **Create Site** section, enter the site pattern, such as `www.company.com`. To enable Oracle Web Cache to match requests to this site, do not add protocol information (`http://` or `https://`) to the host name. In the **Aliases** section, enter the alias name for the site, such as `company.com`. To enable Oracle Web Cache to match requests to this alias, do not add protocol information (`http://` or `https://`) to the host name. **Note:** Do not use the wildcard `*` to represent multiple sites.
Port	Enter the HTTP or HTTPS port number from which Oracle Web Cache is listening for incoming requests.

URL Prefix	To distinguish sites that share the same host name, enter the path prefix of the URLs. Ensure the prefix starts with "/". Do not include the file name or embedded URL parameters in the prefix.
	
	For example, the following URLs share the same site name, but belong to two entirely differently applications potentially hosted on entirely different computers:
	
	`http://www.company.com/portal/page?_pageid=33,4232&_dad=portal`
	`http://www.company.com/um/traffic_cop?mailid=inbox`
	
	These URLs are from completely different applications hosted on the same or different origin server. While the first URL shows an mail 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.

Default Site	Click this option to make this site the default site Oracle Web Cache uses to forward requests without host information.
Compression	Click to instruct Oracle Web Cache to serve cacheable and non-cacheable content compressed to browsers. Not selecting this option means you are instructing Oracle Web Cache to not serve compressed content for this site. See Section 1.2.5 to understand when Oracle Web Cache automatically disables compression. You can disable compression for requests that do not match any site using Oracle Web Cache Manager. See Section 2.11.3.1.

2.11.3.1 Disabling Compression for All Responses

You can disable Oracle Web Cache from compressing all responses.

1. For named sites, deselect the **Compression** option in the Create Site page in Fusion Middleware Control, as described in Section 2.11.3.

2. For undefined sites for requests that do not match any site, use Oracle Web Cache Manager and perform the following tasks:

   a. From Oracle Web Cache Manager, in the navigator frame, select **Origin Servers, Sites, and Load Balancing** > **Site Definitions**. See Section 2.7.2.

      The Site Definitions page displays.

   b. Select the **Undefined Sites** row, and then click **Show/Edit Selected**.

      The Show/Edit Undefined Sites Definition dialog displays.

   c. For the **Site-Wide Compression** element, click **No**.

   d. Click **Submit**.

   e. Click **Apply Changes**.

   f. Restart Oracle Web Cache. See Section 2.13.

2.11.4 Task 4: Map Site Definitions to Origin Servers

After you specify site definitions, you create ordered mappings of sites to origin servers. For an overview of site configuration, see Section 2.2.

If Oracle HTTP Server was installed, the installation process creates a default site-to-server mapping based on the host name and listening port of Oracle HTTP Server.

If you configured multiple origin servers in Section 2.11.2 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 does not occur in the intended manner.

To map sites to origin servers:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.

2. From the **Web Cache** menu, select **Administration** and then **Sites**.

   The Sites page displays.

3. From the **Site-to-Server Mapping** section, click **Create**.

   The Create Site-to-Server Mapping page displays.

4. Configure the **Host Pattern**, **Port Pattern**, and **Prefix** elements:

   a. In the **Host Pattern** field, enter the site pattern, such as `www.company.com`. To enable Oracle Web Cache to match requests to this site, do not add protocol information (`http://` or `https://`) to the host name.

      You can use the wildcard `*` in the **Host Pattern** 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 sites outside a firewall and accessible by a proxy server. For example, `*` can be used to map to proxy server `proxy-host`.

   b. In the **Port Pattern** field, enter the HTTP or HTTPS port number for the Web site from which Oracle Web Cache is listening for incoming requests.

You can use the wildcard * in the **Port Pattern** 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 Section 2.11.2.

5. In the **Origin Servers** section, select the origin servers.

   If you select multiple origin servers, the servers must be of the same type and use the same protocol on their listening port (HTTP or HTTPS). For example, you cannot have a mix of application Web servers and proxy servers.

6. Click **OK** to apply changes and return to the Sites page. It is not necessary to click **Apply** in the Sites page to apply this change.

7. Repeat Steps 3 to 6 for each additional mapping.

8. In the Sites page, use the **Move Up** and **Move Down** features to order the mappings.

   Oracle Web Cache resolves an incoming request first to a site definition, and then to the first matching site-to-origin server mapping. See Section 2.2 for more information about how Oracle Web Cache uses the order of site definitions and site-to-server mappings to match requests..

2.11.5 Task 5: Set Resource Limits and Network Thresholds

For more information about resource limits, see Section 2.3.

To specify caching and network thresholds for Oracle Web Cache:

1. Specify caching thresholds:

   1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.

   2. From the **Web Cache** menu, select **Administration** and then **Resource Limits**.

      The Resource Limits page displays.

   3. In the **Maximum Cached Object Size** field, specify a maximum size of objects to be stored in the cache in kilobytes (KB).

      For more information about the value to enter, see Section 2.3.3

   4. For each cache, in the **Maximum Cache Size** field, enter the amount of memory the Oracle Web Cache requires in megabytes (MB).

      For more information about the value to enter, see Section 2.3.1.

   5. In the **Maximum Incoming Connections** field, enter the maximum number of incoming connections to Oracle Web Cache.

      For more information about the value to enter, see Section 2.3.2.

   6. Click **Apply**.

2. For all the caches, modify the network timeouts:

   a. From Oracle Web Cache Manager, select **Properties** > **Network Timeouts**. See Section 2.7.2.

   b. From the **For Cache** list, select a specific cache.

   c. Select a timeout type and click **Edit Selected**. For more information about the timeouts, see Section 2.3.4.

d. In the Edit dialog for the threshold, modify the value for the **Duration** field or select **Use Default** to use the default value.

   For the keep-alive timeout, if you set the value to 0, the connection to the client is not kept open. In addition, Oracle Web Cache sends the following response-header field in the response:

   ```
   Connection: Close
   ```

   For more information about the value to enter, see Section 2.3.4.

e. Select option **Use for all caches in the cluster** to apply the duration to all caches; deselect the option to apply the change to current cache only.

f. Click **Submit**.

g. Click **Apply Changes**.

   You can always revert to the default values. In the Network Timeouts page, click **Use Defaults**, and then click **Apply Changes** to apply changes.

2.11.6 Task 6: Configure Error Pages

For situations in which there is a network communication error, site busy error, or ESI `<esi:include>` error, applications serve error pages. Rather than burden the origin server with this task, you can configure these pages to be served from Oracle Web Cache.

To configure Oracle Web Cache to serve error pages for a site:

1. Create error pages and place them in the following directory locations:

   ```
   (UNIX) ORACLE_INSTANCE/<instance_name>/config/WebCache/<webcache_name>/files
   (Windows) ORACLE_INSTANCE\<instance_name>\config\WebCache\<webcache_name>\files
   ```

   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 Oracle Web Cache cannot 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.

   For a production environment, modify the defaults or create entirely new error pages to be consistent with other error pages for the site.

2. From Oracle Web Cache Manager, in the navigator frame, select **Origin Servers, Sites, and Load Balancing** > **Error Pages**.

   The Error Pages page appears.

3. Select either ***Default Pages*** or a site name in the table, and then click **Edit**.

   The Edit Error Pages dialog box appears.

4. In the **Network Error Page** field, enter the file name of the error page delivered for network communication problems between Oracle Web Cache and the Web site.

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

5. In the **Site Busy Page** field, enter the file name of the error page delivered when a Web site is saturated with requests.

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

6. In the **ESI Default Fragment** field, enter the file name of the page delivered when Oracle Web Cache cannot 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.

7. Click **Apply Changes**.

   If you selected ***Default Pages***, Oracle Web Cache applies the new settings to all defined sites with the *default page* setting. However, Oracle Web Cache does not apply the new setting to undefined sites. If you selected a specific site in Step 3, Oracle Web Cache applies the new settings to the specific site.

2.11.7 Task 7: Restart Oracle Web Cache

See Section 2.13 for instructions on restarting Oracle Web Cache.

2.12 Creating Session Definitions

You create session definitions for the following features:

- Specify how session requests are served by the cache
- Enable **session binding**, whereby a user session for a particular site is bound to an origin server to maintain state for a period.
- Substitute session information in **session-encoded URLs**

When you enable these features, you must select a session definition.

To create a session definition:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration**, and then **Session Configuration**.

   The Session Configuration page displays.

3. From the **Site** list, select the specific site for which you want to apply this session definition. To create a global session definition that can be applied to any site, select **Global**.

   See Section 2.11.3 to specify additional sites.

4. In the **Session Definitions** section, click **Create**.

   A new row in the table appears.

5. In the **Session Name** field, enter an easy-to-remember unique name for the session.
6. Enter the cookie name in the **Cookie Name** field and the embedded URL parameter in the **URL Post Body Parameters** 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.

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.

---

7. In the **URL Post Body Parameters** field, enter the embedded URL parameter or POST body parameter containing the session information.
8. In the **Default Value** field, enter the default string for Oracle Web Cache to use for the cookie or embedded URL parameter value. Oracle Web Cache uses the default string for those requests without the cookie or parameter information. For these requests, Oracle Web Cache substitutes the session ID information with the default string. The default string defaults to `default`.
9. Click **Apply** to apply changes
10. Restart Oracle Web Cache. See Section 2.13.

For more information about Oracle Web Cache properties requiring session definitions, see:

- Section 3.5 to configure session binding
- Section 6.8.6 to configure session caching rules
- Section 6.8.7 to configure support for session-encoded URLs

2.13 Starting and Stopping Oracle Web Cache

Most configuration changes are static. When you apply static changes, you must restart Oracle Web Cache to apply changes.

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

- Request filtering in both Fusion Middleware Control and Oracle Web Cache Manager
- Setting buffering and verbosity detail level in the Event Logs page (**Logging and Diagnostics** > **Event Logs**) in Oracle Web Cache Manager
- Setting buffering in the Access Logs page (**Logging and Diagnostics** > **Access Logs**) in Oracle Web Cache Manager
- Enabling and disabling of diagnostics information in the HTML response body of an object in the Diagnostics page (**Logging and Diagnostics** > **Diagnostics**) in Oracle Web Cache Manager
- Setting routing to origin servers in the Origin Servers page (**Origin Servers, Sites, and Load Balancing** > **Origin Servers**) in Fusion Middleware Control

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

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

The cache server binary is managed by the webcached executable, and the admin server binary webcachea. These executable reside in the following directories:

```
(UNIX) ORACLE_HOME/webache/bin
(Windows) ORACLE_HOME\bin
```

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

After you configure Oracle Web Cache, restart Oracle Web Cache. To restart Oracle Web Cache, use these tools:

- Use Fusion Middleware Control or opmnctl command-line tool to restart the cache or admin server processes.
- Use Oracle Web Cache Manager to restart the cache server process.

You must restart *both* the cache server and admin server processes if you modified any of these configuration settings:

- Administration port properties (Section 2.11.1.4)
- Trusted subnets (Section 5.3)
- User and group ID information (Section 5.9.1)

2.13.1 Starting and Stopping Using opmnctl

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

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

   ```
   (UNIX) ORACLE_INSTANCE/bin/opmnctl status
   (Windows) ORACLE_INSTANCE\bin\opmnctl status
   ```

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

   ```
   opmnctl status -l
   Processes in Instance: instance1
   ---------------------------------+--------------------+---------+----------+---
   ---------+----------+-----------+------
   ias-component                  | process-type       |     pid | status   |
   uid |  memused |    uptime | ports
   ---------------------------------+--------------------+---------+----------+---
   ---------+----------+-----------+------
   webcache1                        | WebCache-admin     |   11244 | Alive    |
   458185369 |    45852 | 180:27:00 | http_admin:7786
   webcache1                        | WebCache           |   11243 | Alive    |
   458185368 |    72540 | 180:27:00 | http_stat:7787,http_invalidation:7788,https_
   listen:7789,http_listen:7785
   ohs1                             | OHS                |    6077 | Alive    |
   458185358 |   349220 | 180:48:14 | https:9999,https:4443,http:7777
   ```

2. To start, stop, or restart both the admin server (WebCache-admin) and the cache server (WebCache) processes, from the command line, enter:

   ```
   opmnctl startproc|stopproc|restartproc ias-component=component_name
   ```

   To individually start, stop, or restart the admin server (WebCache-admin) and the cache server (WebCache) processes, from the command line, enter:

   ```
   opmnctl startproc|stopproc|restartproc ias-component=component_name
   ```

```
process-type=WebCache-admin

opmnctl startproc|stopproc|restartproc ias-component=component_name
process-type=WebCache
```

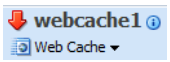
For more information about `opmnctl` for Oracle Web Cache, see the following:

- Section 2.8 for a list of the `opmnctl` commands for Oracle Web Cache
- *Oracle Fusion Middleware Oracle Process Manager and Notification Server Administrator's Guide* to learn more about using `opmnctl`

2.13.2 Starting and Stopping Using the Fusion Middleware Control

To start, stop, or restart Oracle Web Cache from Fusion Middleware Control:

1. Start Fusion Middleware Control. See Section 2.6.1.
2. From the navigation pane, expand the farm and then the **Web Tier** installation type.
3. Select the Oracle Web Cache component, such as **webcache1**.
4. View the target name to determine the status of the cache:



   The arrow reflects the up or down status of the `cache` server process, not the `admin` server process.

   A green up arrow means the following:

   - The `cache` server is running, but the `admin` server process is not running.
   - Both the `cache` server and `admin` server process are running.

   A red down arrow means the following:

   - The `cache` server is not running, but the `admin` server process is running.
   - Both the `cache` server and `admin` server process are not running.

   If the `admin` server process is down, the context pane for the configuration pages displays an error, indicating that configuration is unavailable because the `admin` server process is down.

5. From the Web Cache menu, choose **Control**, then **Start Up**, **Shut Down**, or **Restart**.

   These commands start, stop, or restart both processes, if they have the same up or down status. If the two processes have different up and down statuses, then Fusion Middleware Control starts, stops, or restarts the appropriate process. For example, if the `cache` server process is running, but the `admin` server process is not and you choose **Start Up**, then only the `admin` server is started.

2.13.3 Starting and Stopping Using Oracle Web Cache Manager

Oracle Web Cache Manager enables you to start and stop the `cache` server process. You must use Fusion Middleware Control or `opmnctl` to start, stop, or restart the `admin` server process.

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

1. Start Oracle Web Cache Manager. See Section 2.7.1.

2. In the navigator frame, select **Operations** > **Cache Operations**. See Section 2.7.2.

   The Cache Operations page appears in the right pane.

3. Select the cache, and then 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**.

3

Configuring High Availability Solutions

This chapter describes how to configure and implement high availability solutions using Oracle Web Cache.

This chapter includes the following topics:

3.1 Overview of Origin Server Load Balancing and Failover

You can configure Oracle Web Cache with the application Web servers or proxy servers to which it sends cache misses. Typically, Oracle Web Cache uses application Web servers for internal sites and proxy servers for external sites outside a firewall.

This section covers the following concepts:

For instructions on configuring origin servers, see Section 2.11.2.

3.1.1 Surge Protection

Oracle Web Cache passes requests for non-cacheable, stale, or missing objects to origin servers. To prevent an overload of requests on the origin servers, Oracle 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 Oracle Web Cache rejects the request and serves a site busy error page to the client that initiated the request.

3.1.2 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 Oracle Web Cache cannot serve are passed to the origin servers. Oracle Web Cache balances the load among origin servers by determining the percentage of the available **capacity**, the **weighted available capacity** of each origin server. Oracle 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, Oracle 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, Oracle Web Cache sends the request to the origin server with the most available capacity.

If the load of origin servers is equivalent, Oracle 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.

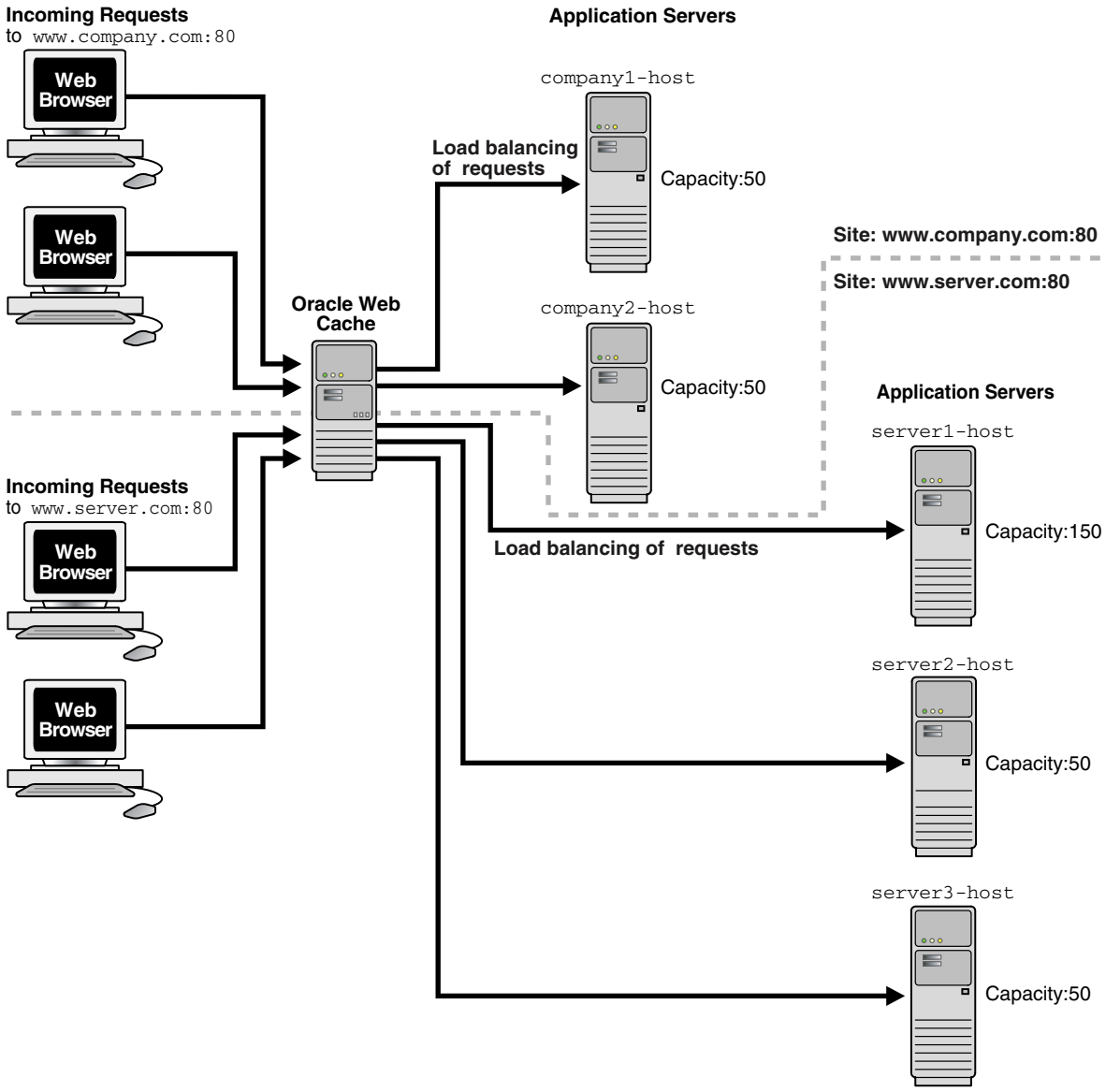
To configure load balancing for a site, set the capacity of each origin server, and create *one* site-to-server mapping that maps *all* the applicable origin servers to the site.

For further information about configuration, see:

- Section 2.11.2 for instructions on specifying capacity
- Section 2.11.4 for instructions on creating site-to-server mappings

Figure 3–1 shows two sites, `www.company.com:80` and `www.server.com:80`. The site `www.company.com:80` is supported by application servers `company-host1` and `company-host2` with capacities of 50 each. The site `www.server.com:80` is supported by application servers `server-host1`, `server-host2`, and `server-host3` with capacities of 150, 50, and 50, respectively.

Figure 3–1 Load Balancing



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

- Oracle Web Cache distributes the requests to `www.company.com:80` between the two application servers using round robin.

  Oracle Web Cache distributes the requests to `company-host1` and `company-host2` between the two application servers so that they maintain an equal load. The first request is sent to `company-host1`. The second request is sent to `company-host2` if `company-host1` is still processing the first request. The third and subsequent requests are sent to the application server that has the highest weighted available capacity.

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

- Oracle Web Cache distributes the requests to `www.server.com:80` between three origin servers using the weighted available capacity percentage.

  The first request to `www.server.com:80` is sent to `server-host1`, because it is the first in the configured list. The second request is sent to `server2-host`, because `server-host1` is still processing the first request and has a weighted available capacity of 99.3 percent and `server-host2` has a weighted available capacity of 100 percent. The third request is sent to `server-host3` 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 `server-host1` because `server-host2` and `server3-host` are still processing requests and have weighted available capacities of 98 percent. The fifth request is sent to `server-host1` because its weighted available capacity is 98.6 percent, which is still greater than `server-host2` and `server-host3`, respectively.

  When the capacities and loads are not equal, Oracle 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, Oracle 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 Oracle 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 Section 3.4 for further information.

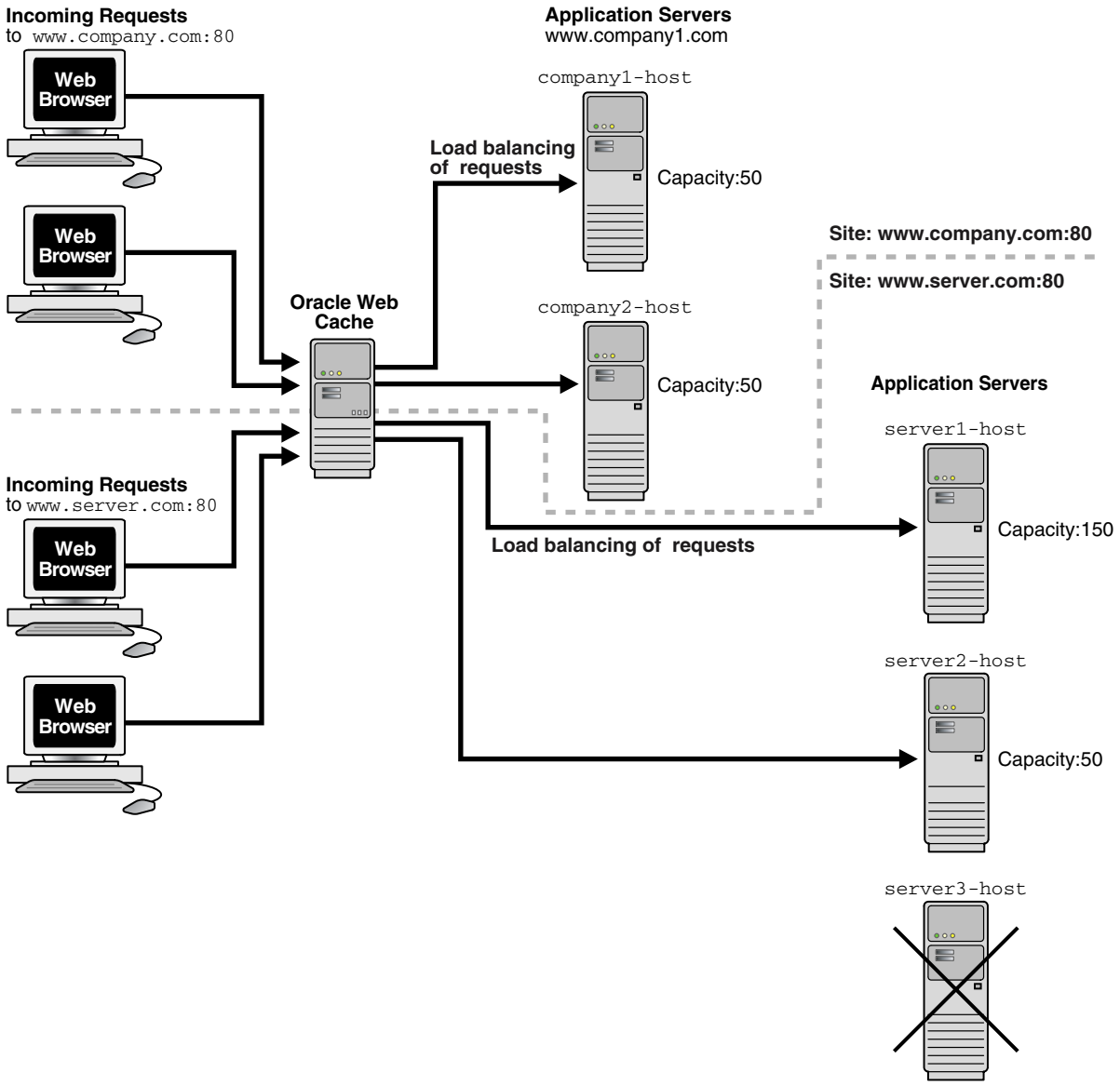
3.1.3 Backend Failover

After a specified number of continuous request failures, Oracle Web Cache considers an origin server as failed. When an origin server fails, Oracle 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, Oracle Web Cache includes it in its weighted available capacity to load balance requests.

For further information about configuring the number of request failures, see Section 2.11.2.

The **failover** feature is shown in Figure 3–2. An outage of `server-host3`, which had a capacity of 50, results in 75 percent of requests being distributed to `server-host1` and 25 percent request being distributed to `server-host2`.

Figure 3–2 Failover



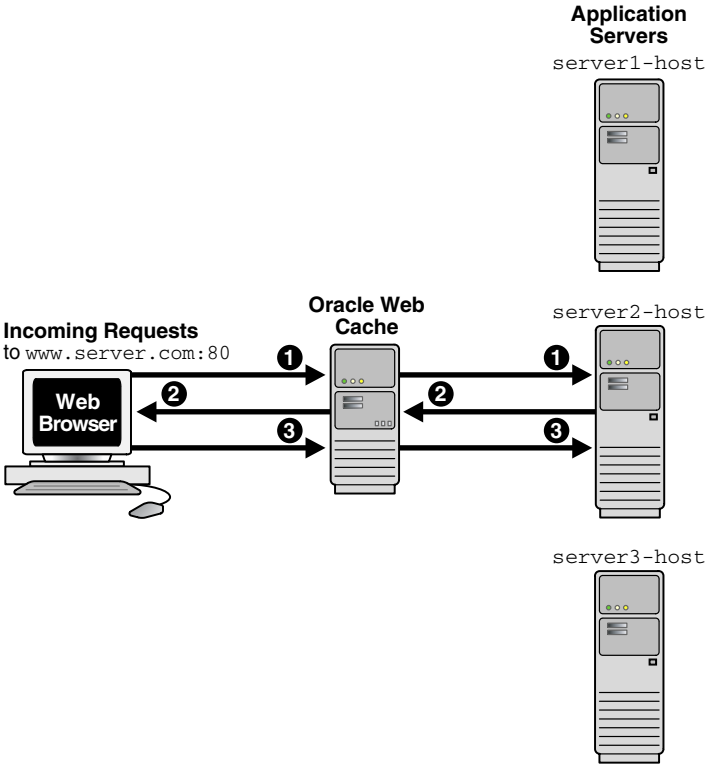
3.2 Overview of Session Binding

You can configure Oracle Web Cache to support **session binding**, whereby a user session for a particular site is bound to an origin server to maintain state for a period. To use 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 Oracle Web Cache in the form of a **session cookie** or an embedded URL parameter. Oracle 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, Oracle Web Cache forwards the request to the origin server that created the user session. Oracle Web Cache binds the user session to that particular origin server.

Figure 3–3 shows how Oracle Web Cache supports objects that use session binding.

Figure 3–3 Session Binding



The steps for how session binding works for requests are as follows:

1. When a request first comes in, Oracle Web Cache uses load balancing to determine to which origin server the request is forwarded. In this example, application 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 Oracle Web Cache in the form of a cookie or an embedded URL parameter.
3. Oracle Web Cache sends subsequent requests for the session to the origin server that established the session, bypassing load balancing. In this example, application server `www.server2.com` handles the subsequent requests.

For instructions on configuring origin servers, see Section 2.12.

If you configure a cache cluster, when you configure session binding, do not select the **Internal-Tracking** mechanism option, as it does not work for cache clusters. The other mechanisms work for cache clusters. See Section 3.6.4 for further information.

Notes:

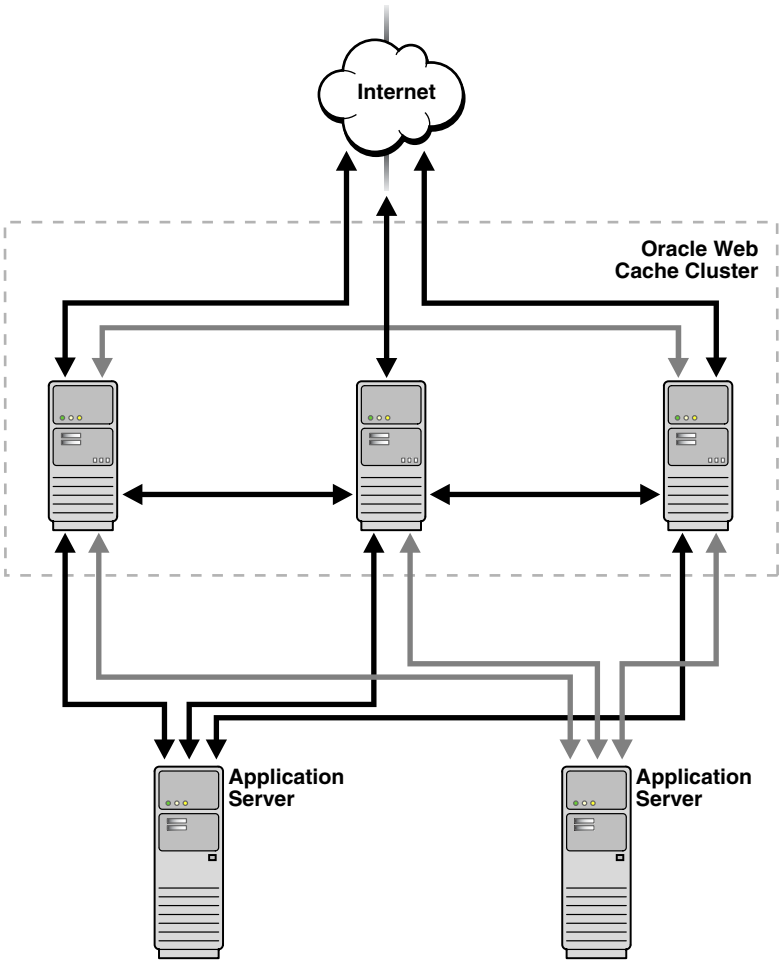
- When a session expires, Oracle Web Cache does not continue to bind the user session to the origin server. Instead, Oracle Web Cache uses load balancing to choose an origin server. To avoid pages being served past the client session expiration time, ensure that the session cookie expires before the origin server expires the client session.
- If an origin server is busy, Oracle Web Cache disables session binding to that origin server.

3.3 Overview of Cache Clusters

In a **cache cluster**, multiple system components of Oracle Web Cache operate as one logical cache. This one logical cache is referred to as the **cache cluster member**. 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 3–4 shows an Oracle 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 3–4 Oracle Web Cache Cluster Architecture



Oracle 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, Oracle 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, Oracle 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.

A cache cluster uses one configuration that is synchronized with 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 Oracle 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, Oracle Web Cache synchronizes the configuration to all members of the cluster. Because adding a new member changes the relative capacity of each Web cache, Oracle 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, Oracle 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 synchronized with 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.

Cache clusters provide the following benefits:

- High availability

  With or without cache clusters, Oracle Web Cache ensures that cache misses are directed to the most available, highest-performing Web server. With cache clusters, Oracle 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. 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 multiple caches. 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 require routing 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, Oracle Web Cache can cache more content, resulting in fewer requests to the application Web server.

- Improved data consistency

  Because Oracle 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 synchronizing configuration changes. See Section 3.6.7 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. Oracle Web Cache distributes those rules throughout the cluster by synchronizing the configuration to each cluster member.

3.4 Overview of High Availability without a Hardware Load Balancer

For environments in which a hardware **load balancer** is not available, you can select to configure the following options:

- Section 3.4.1, "Oracle Web Cache Solely as a Software Load Balancer or Reverse Proxy"
- Section 3.4.2, "Operating System Load Balancing Support"

3.4.1 Oracle Web Cache Solely as a Software Load Balancer or Reverse Proxy

You can configure a special mode of Oracle Web Cache that enables you to use Oracle Web Cache solely as a software load balancer of HTTP traffic or **reverse proxy** to origin servers. This configuration mode is useful to:

- Manage low-volume, departmental, or test Web sites
- Manage traffic in the **DMZ** between a hardware load balancer and an application using Oracle Portal or Oracle Single Sign-On

This mode does not cache *any* content or provide support for the following features:

- Compression: Oracle Web Cache ignores all compression settings.
- ESI: Oracle 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 cache as a load balancer.

You can deploy a single Oracle Web Cache server as a load balancer. However, this deployment makes the Oracle Web Cache server a single point of failure for your application. You can instead configure a cache cluster using multiple Oracle Web Cache servers with operating system load balancing capabilities. Take note of the capacity changes mentioned earlier in this section.

In this mode, you can configure Oracle Web Cache to load balance HTTP traffic in front of an application using ESI or in front of another Oracle Web Cache. The Oracle Web Cache load balancer does not process ESI content or participate in hierarchical caching. For example, a typical Oracle Portal deployment has a built-in Oracle Web Cache used for ESI assembly. For these configurations, do not configure the Oracle Web Cache used for ESI assembly as a load balancer.

If you require other Oracle 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.

For more information, see:

- Section 3.8 for instructions on configuring Oracle Web Cache as a load balancer
- Section 3.9 for instructions on configuring operating system load balancing capabilities
- Section 3.6 for instructions on configuring a cache cluster
- *Oracle Fusion Middleware Enterprise Deployment Guide for Java EE* for instructions on using Oracle Web Cache as reverse proxy for Oracle Portal and Oracle Single Sign-On.

3.4.2 Operating System Load Balancing Support

Certain operating systems provide load balancing support, which can increase the availability of Oracle Web Cache, particularly in cache clusters.

When the operating system detects a failure of one cache, 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 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 met, you could consider using a network load balancer.

Section 3.9 describes how to configure a network load balancer on one operating system.

3.5 Configuring Session Binding

For more information about session binding, see Section 3.2.

To configure session binding, you specify a set of session binding rules, and then apply them to the sites. By default, sites are configured to use a default rule. You can use the default rule or select another rule customized for the site.

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 default rule, and instead specify session binding settings for the site.

To configure session binding:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the Web Cache menu, select **Administration** and then select **Session Configuration**.

   The Session Configuration page displays.
3. From the **Site** list, select the site to create customized session-bindings.

   Select **Global** to specify default settings for requests which do not match any defined site. If you do not specify customized session-binding settings for a site, then you can click the **Use global settings** option to apply the settings you specify for **Global**. The default selection for the Global selection is the **Disable session binding**. You change the default setting by selecting **Global** from the **Site** list and selecting on of the other session-binding selections.
4. Create a session definition in the **Session Definitions** table. See Section 2.12.

   - **Use global settings:** Select this option to apply the session-binding settings you configured for the **Global** selection from the **Site** list.

   By default, Oracle Web Cache disables session binding for all requests. The default selection for **Global** is the **Disable session binding** option. When you first create a site, it is set by default to use the global session binding settings

   - **Disable session binding:** Select this option to disable session binding. This selection is the default for the Global site. You change the default setting by selecting **Global** from the **Site** list and selecting on of the other session-binding selections.

   - **Cookie based session binding with any Set-Cookie:** Select this option if the client supports cookies and your origin server uses one or more cookies for session state. Oracle Web Cache uses its own cookie to track a session. This cookie, which contains routing information, is maintained between Oracle Web Cache and the client browser. The client/server binding is initiated by the first cookie that the application server sends to the client.

   - **Bind using a session:** Select this option to enable binding for a specific session, and then perform the following steps:

   a. From the **Session Name** list, select a session to enable binding for a specific session.

   b. From the **Session Binding Mechanism** list, select a binding mechanism for the selected session definition:

      - **Cookie Based:** Select if the client supports cookies. Oracle Web Cache uses its own cookie to track a session. This cookie, which contains routing information, is maintained between Oracle Web Cache and the client browser.

- **Session Binding IAS:** Select if the application is based on OC4J. Oracle 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 Oracle Web Cache. Oracle 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 also bind at the load balancer layer.

5. Click **Apply** to submit changes.
6. Restart Oracle Web Cache. See Section 2.13.

3.6 Configuring a Cache Cluster for Caches Using the Same Oracle WebLogic Server

To increase the availability and scalability of your Web site, you can configure multiple instances of Oracle Web Cache to run as members of a cache cluster. For more information about cache clusters, see Section 3.3.

To configure a cache cluster, you configure two or more Oracle Web Cache instances as cache cluster members, and specify properties for the cluster.

A cache cluster uses one configuration that is synchronized 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.

This section contains the following topics to aid you in configuring a cache cluster in a environment in which all the caches are associated with the same Oracle WebLogic Server. These instruction explain how to configure a cluster using Fusion Middleware Control, which requires all the cache members to use the same Oracle WebLogic Server:

- Section 3.6.1, "Configuration Prerequisites"
- Section 3.6.2, "Understanding Failover Threshold and Capacity Settings"
- Section 3.6.3, "Task 1: Add Caches to the Cluster and Configure Properties"
- Section 3.6.4, "Task 2: Enable Tracking of Session Binding"
- Section 3.6.5, "Task 3: Synchronize the Configuration to Cluster Members"

In addition, see the following information about configuring clusters:

- Section 3.6.6, "Removing a Cache Member from a Cluster"
- Section 3.6.7, "Configuring Administration and Invalidation-Only Clusters"

If you have want to configure a cache cluster for a configuration in which the caches are associated with different Oracle WebLogic Servers, follow the instructions in Section 3.7 to use Oracle Web Cache Manager to configure the cluster.

3.6.1 Configuration Prerequisites

Because a cache cluster contains two or more instances of Oracle Web Cache, you must have two or more instances of Oracle Web Cache installed on one or more nodes before you configure a cache cluster. The instances must be the same version of Oracle

Web Cache. In addition, the respective passwords for the Oracle Web Cache administrator, and the invalidator user, `invalidator`, must be the same across the cluster members. If they are different, you must connect to the cache's `admin` server and modify the administration password, as described in Section 5.2.

3.6.2 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:

- Section 3.6.2.1, "Failover Threshold for the Cache Cluster"
- Section 3.6.2.2, "Capacity for Cache Cluster Members"

3.6.2.1 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 Oracle Web Cache considers another cache cluster member to have failed. In other words, Oracle Web Cache consecutively retries a failed member for certain number of times, before considering the cache-member as down. The number of times Oracle Web Cache is allowed to retry is termed as failover threshold.

Oracle 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 500 Internal Server Error, 502 Bad Gateway, 503 Service Unavailable, or 504 Gateway Timeout, or less than 100.

For each failed request, Oracle 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, Oracle Web Cache resets the failure counter.

When the failover threshold is met, Oracle Web Cache considers the cache cluster member to have failed. Oracle 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, Oracle Web Cache starts polling the cache cluster member. It does this by sending requests to the ping URL you specify. When Oracle 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.

3.6.2.2 Capacity for Cache Cluster Members

When you configure a cache cluster member, you specify capacity for that member.

Oracle 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, Oracle Web Cache assigns a percentage of the ownership array to the cluster member, indicating how much of the cached content is to 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, Oracle 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 Incoming Connections** setting.

After you have a better idea of an application'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 is distributed equally to all the cache cluster members.
2. Ownership of content is 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:
  - 100 for production environments
  - 30 for test environments
  - 0 for invalidation-only clusters

  When the capacity increases, the number of file descriptors needed by Oracle Web Cache also increases.

  See Section 3.6.7 for further information about invalidation-only clusters.
- *max_incoming_connections* is the **Maximum Incoming Connections** setting from the Resource Limits page of Fusion Middleware Control.

- *cacheable_misses%* is the percentage of requests for cacheable objects that were not served directly by Oracle Web Cache, but were served by Oracle Web Cache after it fetched the content from the origin server.

  You can find the **Cacheable Misses** setting in the Web Cache Statistics page of Fusion Middleware Control.

For example, assume a cache cluster with four members. If Oracle 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 does not receive requests from other cluster members. However, that cluster member does forward requests to other cluster members, the owners of the content. If you assign a capacity of 0 to *all* cluster members, Oracle Web Cache does not forward requests between cluster members. Even when capacity is set to 0, you can still synchronize the configuration and Oracle Web Cache can automatically propagate invalidation requests to cluster members.

3.6.3 Task 1: Add Caches to the Cluster and Configure Properties

Before you add a cache to the cluster, ensure the conditions described in Section 3.6.1 are met.

To add cache members to a cluster with Fusion Middleware Control:

1. Navigate to the Web Cache Home page in Fusion Middleware Control for an Oracle Web Cache instance. See Section 2.6.2.
2. From the Web Cache menu, select **Administration** and then select **Cluster**.

   The Cluster page displays.
3. For each cache member you want to add:
   a. Click **Add**.
   b. In the **Component** field, enter the name of the cache member.

      The **Capacity** field is auto-filled with a default value. You can modify this value. See Section 3.6.2 for more information about capacity.
4. In the **Failover Threshold** field, enter the number of allowed consecutive request failures before Oracle Web Cache considers another cache cluster member to have failed.

   The default is five failures.

   See Section 3.6.2 for further information about this field.
5. In the **Ping URL** field, enter the URL that cache cluster members uses 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 Frequency** 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. Click **Apply**.

3.6.4 Task 2: 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.

For the Oracle Web Cache you established properties for in Section 3.6.3, configure **session binding** with a session binding mechanism of **Cookie Based** or **Session Binding IAS**. Do not use the **Internal-Tracking** option, as it does not work for cache clusters.

To configure session binding with the Cookie-based mechanism, see Section 3.5.

3.6.5 Task 3: Synchronize the Configuration to Cluster Members

When you modify the cluster and apply changes, Oracle 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 synchronize the configuration and restart the cluster members.

To synchronize configuration from a newly-added cache member to the other caches in the cluster with Fusion Middleware Control:

1. Navigate to the Web Cache Home page in Fusion Middleware Control for the Oracle Web Cache you established properties for in Section 3.6.3.
2. From the Web Cache menu, select **Administration** and then select **Cluster**.

   The Cluster page displays.
3. Select the other cache members in the cluster, click **Synchronize**.
4. Select all the cache members, select an interval for staggering the time that operation begins on the cache sand click **Start Up**.

The cache cluster is ready to use.

3.6.6 Removing a Cache Member from a Cluster

To remove a cache-member from a cluster, you must not only ensure 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 with Fusion Middleware Control:

1. Navigate to the Web Cache Home page in Fusion Middleware Control for an Oracle Web Cache instance, but *not* the cache to remove from the cluster. See Section 2.6.2.
2. From the Web Cache menu, select **Administration** and then select **Cluster**.

   The Cluster page displays.
3. Select the cache you want to remove and click **Delete.**
4. Select the other cache members in the cluster, click **Synchronize**.

5. With the other caches still selected, 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. You remedy this situation in the next steps.

6. Navigate to the Web Cache Home page in Fusion Middleware Control of the cache you removed from the cluster.

7. From the Web Cache menu, select **Administration** and then select **Cluster**.

   The Cluster page displays with all the member of the cache.

8. Select a cache except the current one, and click **Delete**. Repeat until only the current cache remains in the **Cluster Members** list.

9. Click **Restart**.

3.6.7 Configuring Administration and Invalidation-Only Clusters

You can configure a cluster that supports synchronizing 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 synchronized among 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 Section 3.6.3 and Section 3.6.4, with the following exceptions:
   - For each cluster member, set the capacity to 0.
   - In the **Cluster Properties** section, select option **Invalidation requests sent to any cluster member will be propagated to all cluster members**.
2. Synchronize the configuration to all cluster members, as described in Section 3.6.5.

3.7 Configuring a Cache Cluster for Unassociated Caches or Caches Using Different Oracle WebLogic Servers

This section contains the following topics to help you in configuring a cache cluster in a configuration in which all unassociated caches are using different Oracle WebLogic Servers. These instruction explain how to configure a cluster using Oracle Web Cache Manager.

- Section 3.7.1, "Task 1: Configure Cache Cluster Settings"
- Section 3.7.2, "Task 2: Add Caches to the Cluster"
- Section 3.7.3, "Task 3: Enable Tracking of Session Binding"
- Section 3.7.4, "Task 4: Synchronize the Configuration to Cluster Members"

In addition, see the following information about configuring clusters:

- Section 3.7.5, "Removing Caches from a Cluster"
- Section 3.7.6, "Configuring Administration and Invalidation-Only Clusters"

To configure a cache cluster for a configuration in which the caches are associated with same Oracle WebLogic Server, follow the instructions in Section 3.6 to use Fusion Middleware Control to configure the cluster.

3.7.1 Task 1: Configure Cache Cluster Settings

To configure settings for a cache cluster with Oracle Web Cache Manager:

1. From Oracle Web Cache Manager, in the navigator frame, select **Properties** > **Clustering**. See Section 2.7.2.

   The Clustering page appears. The General Cluster Information section displays the default clusterwide values for failover and invalidation synchronization. The Cluster Members table displays the current cache (the cache to which you are connected) as the only cluster member. Oracle 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 Oracle Web Cache considers another cache cluster member to have failed.

   The default is five failures.

   See Section 3.6.2 for further information about this field.

5. In the **Ping URL** field, enter the URL that cache cluster members uses 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 you want all invalidation requests from any cache cluster member to be synchronized with 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 Oracle 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 Oracle Web Cache is installed. Usually, you do not have to modify this field.

12. By default, the **Oracle Home** field contains the file specification for the Oracle home in which Oracle Web Cache is installed. Usually, you do not have 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 Oracle Web Cache can sustain.

    See Section 3.6.2 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 multiple Oracle Web Cache system components in the cluster.

3.7.2 Task 2: Add Caches to the Cluster

Before you add a cache to the cluster, ensure the conditions described in Section 3.6.1 are met.

To add another cache to the cluster with Oracle 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 Oracle Web Cache can sustain.

   See Section 3.6.2 for further information about this field.
8. Click **Submit.**

   The cache is now part of the cluster and is listed in the Cluster Member table.
9. To add more Oracle 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**.

Oracle Web Cache adds the cache-specific information from the new cache cluster members to the cluster configuration.

You can add more Oracle 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**.

3.7.3 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. Oracle 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.

To configure session binding with the Cookie-based mechanism, see Section 3.5.

3.7.4 Task 4: Synchronize the Configuration to Cluster Members

When you modify the cluster and apply changes, Oracle 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 synchronize the configuration and restart the cache server process of the cluster members.

To synchronize the configuration to new cluster members with Oracle 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 synchronized.

2. Synchronize 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 synchronization.)

   c. Click **Propagate**.

   (Alternatively, you can synchronize 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 must 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.

3.7.5 Removing Caches from a Cluster

To remove a cache from a cluster, you must not only ensure 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 with Oracle Web Cache Manager:

1. Enter the URL for the Oracle Web Cache Manager of a cache in cluster, but *not* the cache to remove from the cluster.
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 Oracle Web Cache Manager main window, click **Apply Changes**.
5. Synchronize 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 synchronized with 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 Oracle Web Cache Manager of the cache you removed from the cluster.
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 Oracle Web Cache Manager main window, click **Apply Changes**.
11. In the navigator frame, select **Operations > Cache Operations**.
12. Select the cache and click **Restart**.

3.7.6 Configuring Administration and Invalidation-Only Clusters

You can configure a cluster that supports synchronizing 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 synchronized among 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 Section 3.7.1 and Section 3.7.2, 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. Synchronize the configuration to all cluster members, as described in Section 3.7.4.

3.8 Configuring Oracle Web Cache as a Software Load Balancer

For an overview of high availability without a hardware load balancer, see Section 3.4.

To configure a single Oracle Web Cache server as a software load balancer:

1. Use a text editor to open `webcache.xml`, located in:

   ```
   (UNIX) ORACLE_INSTANCE/<instance_name>/config/WebCache/<webcache_name>
   (Windows) ORACLE_INSTANCE\<instance_name>\config\WebCache\<webcache_name>
   ```

2. Locate the `CACHE` element.
3. Add the `ROUTINGONLY` attribute to the `CACHE` element. For example:

   ```
   ...
   <CACHE WCDEBUGON="NO" CHRONOSONPERNODE="NO" CAPACITY="301" VOTES="1"
   INSTANCENAME="instance_name" COMPONENTNAME="component_name"
   ORACLEINSTANCE="instance" HOSTNAME="web_cache_host_name"
   ORACLEHOME="directory" NAME="web_cache_name"
   ROUTINGONLY="YES">
   ...
   ```

4. Save `webcache.xml`.
5. Restart Oracle Web Cache with the following command:

   ```
   opmnctl restartproc ias-component=component_name
   ```

   This executable is found in the following directory:

   ```
   (UNIX) ORACLE_INSTANCE/bin
   (Windows) ORACLE_INSTANCE\bin
   ```

6. Verify Oracle Web Cache is running in the load balancer mode from the Oracle Web Cache Manager by verifying the following status message displays beneath the **Apply Changes** and **Cancel Changes** buttons:

   ```
   Web Cache running in Routing Only Mode with current configuration
   ```

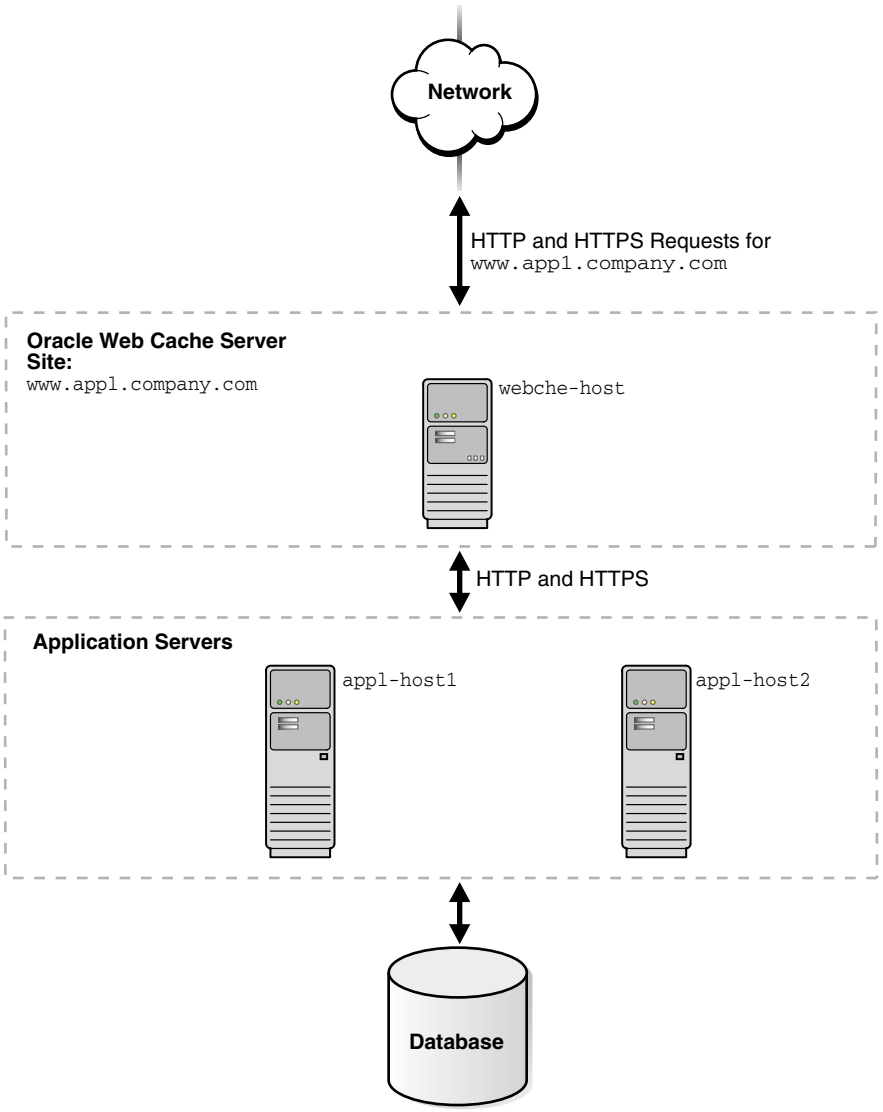
   Fusion Middleware Control does not provide an equivalent verification status.

7. Configure origin servers, as described in Section 2.11.2.
8. Create site definitions and map them to the origin servers, as described in Section 2.11.3 and Section 2.11.4.

9. If your application deployment requires session stickiness, enable session binding. See Section 2.12.

Consider the topology depicted in Figure 3–5.

Figure 3–5 Deploying Oracle Web Cache as a Load Balancer



To configure this topology:

1. Register the IP address of the Oracle Web Cache server `webche-host` with `www.app1.company.com`.
2. Configure the Oracle 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`.

3.9 Configuring Microsoft Windows Network Load Balancing

For an overview of high availability without a hardware load balancer, see Section 3.4.

On certain Microsoft 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 running Oracle 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

4

Configuring Request Filtering

This chapter introduces the request filters provided by Oracle Web Cache and explains how you can enable them to protect against common HTTP request attacks.

This chapter includes the following topics:

4.1 Introduction to Request Filtering

Oracle Web Cache provides request filters to filter incoming HTTP or HTTPS requests to configured sites on the origin server.

Request filtering aids administrators in controlling access to Web sites:

- The planting of malicious code within the Web site that, when executed by a user, steals the user's identity or personal information
- Attacks that try to exploit software vulnerabilities in the site that enable the attacker to execute arbitrary code on the application server.
- Attacks that try to render a Web site unusable by bombarding it with extremely high volumes of bogus requests that effectively consume the application server resources or bandwidth thereby preventing access for other users.

In addition, request filtering controls which clients and requests are allowed to access to a Web site or certain parts of a Web site.

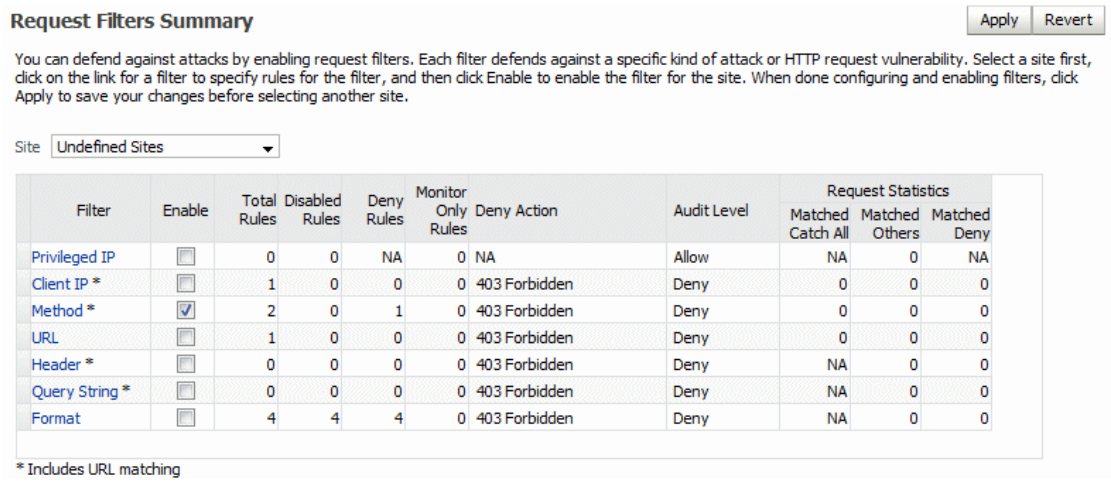
To defend against Web site attacks, you can enable a series of filters that each request must pass through before being processed. Each filter is composed of customizable rules that can either identify the requests to allow or deny.

You can configure filters and filter rules for specific sites or undefined sites. Oracle Web Cache directs client requests that do not match a defined site to the request filters configured for **Undefined Sites**.

The requests filters are processed in the order presented in the Request Filter Summary page. To access this page:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Request Filters**.

   The Request Filters Summary page displays.



Request Filters Summary

Apply | Revert

You can defend against attacks by enabling request filters. Each filter defends against a specific kind of attack or HTTP request vulnerability. Select a site first, click on the link for a filter to specify rules for the filter, and then click Enable to enable the filter for the site. When done configuring and enabling filters, click Apply to save your changes before selecting another site.

Site: Undefined Sites

Filter	Enable	Total Rules	Disabled Rules	Deny Rules	Monitor Only Rules	Deny Action	Audit Level	Request Statistics: Matched Catch All	Matched Others	Matched Deny
Privileged IP	☐	0	0	NA	0	NA	Allow	NA	0	NA
Client IP *	☐	1	0	0	0	403 Forbidden	Deny	0	0	0
Method *	☑	2	0	1	0	403 Forbidden	Deny	0	0	0
URL	☐	1	0	0	0	403 Forbidden	Deny	0	0	0
Header *	☐	0	0	0	0	403 Forbidden	Deny	NA	0	0
Query String *	☐	0	0	0	0	403 Forbidden	Deny	NA	0	0
Format	☐	4	4	4	0	403 Forbidden	Deny	NA	0	0

* Includes URL matching

You select an individual filter from the **Filter** column, and specify individual rules for the filter. When configuring rules, you order the rules based on the order you want Oracle Web Cache to match requests. When ordering caching rules, give allow rules a higher priority than deny rules.

After configuring rules for a filter and enabling or disabling the rules, you return to the Request Filters Summary page to enable the filters. If you do not click **Enable** for a filter, then you are disabling the rule, which means Oracle Web Cache ignores any configured rules for that filter.

4.2 Types of Request Filters

Oracle Web Cache provides the following filters, each designed to focus on a particular type of HTTP request vulnerability.

- Privileged IP
- Client IP
- Method
- URL
- Header

- Query String
- Format

The privileged IP filter permits allow-only rule; the header, query string, and format filters permit deny-only rules; and the client IP, method, and URL filters permit both allow and deny rules. Because the list of rules in the header, query string, and format filters are independent of each other, permitting allow rules could result in the skipping of other deny rules. Therefore, these filters only permit deny rules.

Privileged IP

The privileged IP filter enables Oracle Web Cache to bypass the other request filters. You use this filter to allow specified privileged IP addresses access.

Client IP

The client IP filter allows or denies site access to specific IP addresses.

It enables Oracle Web Cache to restrict access to a site URL prefix within the site to only certain IP addresses. This filter restricts clients from certain IP addresses from launching attacks on a system. Not restricting access could allow clients access to the application or to areas of the site that contain sensitive information. An attacker from a certain IP address can continue making malicious attacks if Oracle Web Cache does not deny access.

You can configure a black list by denying requests if the IP address and URL match or a white list if the IP address and URL match.

Method

The method filter allows or denies site access based on the HTTP request method. For example, if only GET and POST methods are allowed, Oracle Web Cache would refuse all other requests.

This filter protects against clients attempting to read restricted files or modifying files using various HTTP methods. In addition to the HTTP request method, you can configure a URL to limit the rule to only requests that match the method and the specified URL.

URL

The URL filter allows or denies site access based on a URL.

This filter protects against Internet attacks to an application server through a specific URL.

Header

The header filter denies site access based on HTTP header values. In addition to the HTTP header value, you can configure a URL to limit the rule to only requests that match the header value and the specified URL.

Incoming requests matching the HTTP header and URL are compared to the expression in the rule. The expression can be either a substring or a regular expression. For both substring and regular expression comparisons, a rule can deny requests in which the request's header value matches the rule's value expression.

This filter protects against clients attempting to break into an application by manually creating header values and clients submitting unwanted content in header values.

Query String

The query string filters denies site access based on query string parameters. For a POST request, Oracle Web Cache checks both the query string, if is present, and the POST body. In addition to the query string, you can configure a URL to limit the rule to only requests that match the query string and the specified URL.

Incoming requests matching the query string and URL are compared to the expression in the rule. The expression can be either a substring or a regular expression. For both substring and regular expression comparisons, a rule can deny requests in which the request's query string matches the rule's value expression.

This filter protects against clients attempting to break into a site by manually manipulating the query string parameters and values and clients submitting unwanted content within parameter values.

Format

The format filter denies site access based on the format of the HTTP request. This filter checks for embedded null byte characters, strict encoding and valid Unicode, and double URL encoding. Oracle Web Cache checks the format for each enabled type and denies the request if the format is invalid.

This filter checks the components of the URL, including the path, filename, query string, and for POST requests, the request entity body. It protects against hackers attempting to disrupt a Web application by either sending a request which is not well formed or sending characters not expected to be in the URL.

4.3 About Learned Rules

Oracle Web Cache automatically creates learned rules for the method and URL filters. You can then choose to activate these learned rules.

Client requests that match the filter's Catch All rule are evaluated to see if there is some commonality to them that might warrant a new rule. These common patterns are shown as learned rules. You can then chose to activate or ignore these learned rules. After a rule is activated in the configuration, you can select to enable or disable it just like any other rule. Even if you select not to activate learned rules, Oracle Web Cache continues to collect and evaluate all common patterns for requests that fall into the Catch All rule.

See Section 4.7.1 and Section 4.8.1 to enable learned rules.

4.4 About the Monitor Only Mode

When you configure rules for the filters, you can select the **Monitor Only** option. When you enable this option for a rule, Oracle Web Cache treats the rule as if it was disabled. However, Oracle Web Cache tracks matches in the statistics and writes them to the event log (if verbosity is set to TRACE or higher) and to the audit log if audit logging is enabled for the match action.

When monitoring is enabled, requests are allowed, so you can examine results in the **Request Statistics** section. When you disable **Monitor Only** for a deny rule, the deny action is enforced. You typically set **Monitor Only** on to see the match activity of the rule. When results are expected, then disable **Monitor Only** to enforce the rule's action.

4.5 Configuring Rules for the Privileged IP Filter

The privileged IP request filter enables Oracle Web Cache to bypass all request filters for certain privileged IP addresses. Any request from a privileged IP address does not pass through the other request filters.

See Section 4.2 for further information about the privileged IP request filter.

To configure the privileged IP request filter:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Request Filters**.

   The Request Filters Summary page displays.
3. From the **Site** list, select the site to apply the filter. See Section 2.11.3 and Section 2.11.4 to create additional sites.

   You can configure filters and filter rules for specific sites or **Undefined Sites**. Oracle Web Cache directs client requests that do not match a defined site to the request filters configured for **Undefined Sites**.
4. Click the **Privileged IP** link.

   The Privileged IP Request Filter page displays.
5. From the **Audit** list, select the level of action for Oracle Web Cache to include in the audit log for the request filter.
6. Create a new rule:
   a. Click **Create** to create a row in the table.
   b. In the **IP Address** field, enter the IP address, either as an IP version 4 or IP version 6 address mask of the client.

      See Section 2.5 for examples of IP addresses.
   c. Click the **Enable** check box to enable the rule; deselect the check box to disable the rule temporarily without losing the rule definition.
   d. Click the **Monitor Only** check box to see the match activity of the rule without enforcing the rule.

      When results are expected, then disable **Monitor Only** to enforce the rule. See Section 4.4 for further information about the **Monitor Only** option.
   e. Click **Apply** to save the rule settings.
7. Perform Step 6 for any additional rules.
8. Use the **Move Up** and **Move Down** icons to change the order in which the rules are matched against requests.

   The order of the rules is important. Oracle Web Cache matches higher priority rules first.
9. Click the **Request Filters Summary** breadcrumb at the top of the page, or from the **Web Cache** menu, select **Administration** and then **Request Filters** to navigate back to the Request Filters Summary page.
10. In the **Privileged IP** row, click **Enable** to enable the filter.

    If you do not click **Enable**, Oracle Web Cache ignores any configured filter rules for this filter.

11. Click **Apply** to save the configuration for the request filter.

4.6 Configuring Rules for the Client IP Request Filter

This client IP request filter restricts application access to specific IP addresses or range of IP addresses. Not restricting access enables access to restricted information and potential attackers from particular IP addresses.

See Section 4.2 for further information about the client IP request filter.

To configure rules for the client IP request filter:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Request Filters**.

   The Request Filters Summary page displays.
3. From the **Site** list, select the site to apply the filter. See Section 2.11.3 and Section 2.11.4 to create additional sites.

   You can configure filters and filter rules for specific sites or **Undefined Sites**. Oracle Web Cache directs client requests that do not match a defined site to the request filters configured for **Undefined Sites**.
4. Click the **Client IP** link.

   The Client IP Request Filter page displays.
5. From the **Audit** list, select the level of action for Oracle Web Cache to include in the audit log for the request filter.
6. From the **Response to deny** list, select the HTTP response for Oracle Web Cache to return to browsers for requests that are denied by this request filter.

   The **Close Connection** option does not return any HTTP responses. It just closes the connection.
7. Create a new rule:
   a. Click **Create** to create a row in the table.
   b. In the **IP Address** field, enter the IP address, either as an IP version 4 or IP version 6 address mask of the client.

      See Section 2.5 for examples of IP addresses.
   c. Click the **Enable** check box to enable the rule; deselect the check box to disable the rule temporarily without losing the rule definition.
   d. In the **URL** field, based on the **URL Type** you select, enter an optional URL string. If no URLs are specified, then all requests are checked. It is equivalent to specifying a URL with a prefix `/`.

      - **Path Prefix:** Enter the path prefix of the objects. Start the path with `/`; do not start the path with `http://`*`host_name`*`:`*`port`*`/`. 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 (\).

      - **File Extension:** Enter the file extension. Because Oracle Web Cache internally starts the file extension with a period (.), it is not necessary to enter it.

- **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.

e. From the **URL Type** list, select an option to determine how the rule's URL is compared to the request's URL:

- **Path Prefix:** Select to allow or deny access to requests matching a path prefix.

- **File Extension:** Select to allow or deny access to requests matching a particular file extension.

- **Regular Expression:** Select to allow or deny access to requests matching regular expression syntax.

f. Click the **Case Insensitive Match** check box to match requests regardless of the case. If you do not select this check box, the rule bases the match on the case.

g. Click the **Allow** check box for Oracle Web Cache to allow requests matching the IP address and URL fields; deselect the check box for Oracle Web Cache to deny requests matching the IP address and URL fields.

h. Click the **Monitor Only** check box to see the match activity of the rule without enforcing the rule.

When results are expected, then disable **Monitor Only** to enforce the rule. See Section 4.4 for further information about the **Monitor Only** option.

i. Click **Apply** to save the rule settings.

8. Perform Step 7 for any additional rules.
9. Modify the **Catch All** rule, keeping in mind it is applied to all requests that do not match a defined rule.

Oracle recommends creating allow rules, followed by a **Catch All** deny rule.

10. Use the **Move Up** and **Move Down** icons to change the order in which the rules are matched against requests.

The order of the rules is important. Oracle Web Cache matches higher priority rules first.

11. Click the **Request Filters Summary** breadcrumb at the top of the page, or from the **Web Cache** menu, select **Administration** and then **Request Filters** to navigate back to the Request Filters Summary page.
12. In the **Client IP** row, click **Enable** to enable the filter.

If you do not click **Enable**, Oracle Web Cache ignores any configured filter rules for this filter.

13. Click **Apply** to save the configuration for the request filter.

4.7 Configuring Rules for the Method Request Filter

The method request filter enables Oracle Web Cache to restrict access based on the HTTP request method.

See Section 4.2 for further information about the method request filter.

To configure rules for the method request filter:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.

2. From the **Web Cache** menu, select **Administration** and then **Request Filters**.

   The Request Filters Summary page displays.

3. From the **Site** list, select the site to apply the filter. See Section 2.11.3 and Section 2.11.4 to create additional sites.

   You can configure filters and filter rules for specific sites or **Undefined Sites**. Oracle Web Cache directs client requests that do not match a defined site to the request filters configured for **Undefined Sites**.

4. Click the **Method** link.

   The Method Request Filter page displays.

5. From the **Audit** list, select the level of action for Oracle Web Cache to include in the audit log for the request filter.

6. From the **Response to deny** list, select the HTTP response for Oracle Web Cache to return to browsers for requests that are denied by this request filter.

   The **Close Connection** option does not return any HTTP responses. It just closes the connection.

7. In the **Defined Rules** section, create a new rule:

   a. Click **Create** to create a row in the table.

   b. In the **Method** field, enter the HTTP request method, such as GET, POST, or PUT.

   c. Click the **Enable** check box to enable the rule; deselect the check box to disable the rule temporarily without losing the rule definition.

   d. In the **URL** field, based on the **URL Type** you select, enter an optional URL string. If no URLs are specified, then all requests are checked. It is equivalent to specifying a URL with a prefix `/`.

      - **Path Prefix:** Enter the path prefix of the objects. Start the path with `/`; do not start the path with `http://`*`host_name`*`:`*`port`*`/`. 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 (\).

      - **File Extension:** Enter the file extension. Because Oracle Web Cache internally starts the file extension with a period (.), it is not necessary to enter it.

      - **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.

   e. From the **URL Type** list, select an option to determine how the rule's URL is compared to the request's URL:

      - **Path Prefix:** Select to allow or deny access to requests matching a path prefix.

      - **File Extension:** Select to allow or deny access to requests matching a particular file extension.

      - **Regular Expression:** Select to allow or deny access to requests matching regular expression syntax.

   f. Click the **Case Insensitive Match** check box to match requests regardless of the case. If you do not select this check box, the rule bases the match on the case.

g. Select the **Allow** check box for Oracle Web Cache to allow requests matching the method and URL fields; deselect the check box for Oracle Web Cache to deny requests matching the method and URL fields.

h. Click the **Monitor Only** check box to see the match activity of the rule without enforcing the rule.

When results are expected, then disable **Monitor Only** to enforce the rule. See Section 4.4 for further information about the **Monitor Only** option.

i. Click **Apply** to save the rule settings.

8. Perform Step 7 for any additional rules. You can also add learned rules, as described in Section 4.7.1.

9. Modify the **Catch All** rule, keeping in mind it is applied to all requests that do not match a defined rule.

Oracle recommends creating allow rules, followed by a **Catch All** deny rule.

10. Use the **Move Up** and **Move Down** icons to change the order in which the rules are matched against requests.

The order of the rules is important. Oracle Web Cache matches higher priority rules first.

11. Click the **Request Filters Summary** breadcrumb at the top of the page, or from the **Web Cache** menu, select **Administration** and then **Request Filters** to navigate back to the Request Filters Summary page.

12. In the **Method** row, click **Enable** to enable the filter.

If you do not click **Enable**, Oracle Web Cache ignores any configured filter rules for this filter.

13. Click **Apply** to save the configuration for the request filter.

4.7.1 Activating Learned Rules for the Method Request Filter

See Section 4.3 for further information about how learned rules are collected from the **Catch All** rule. You can add learned rules to the method request filter.

To enable learned rules for the method request filter:

1. Navigate to the Method Request Filter page.

2. In the **Learned Rules** section, if you see that some learned rules have been suggested, monitor the statistics for these rules by watching this page for awhile. When you decide that one or more learned rules make sense, proceed.

If no learned rules display under **Catch All Rule**, then there are no learned rules.

3. Select the row, and click **Add Rule to Defined Rules** to activate the rule and move it the **Defined Rules** section.

4. Use the **Move Up** and **Move Down** icons to change the order in which the rules are matched against requests.

5. Click **Apply** to save the configuration for the request filter.

4.8 Configuring Rules for the URL Request Filter

The URL request filter enables Oracle Web Cache to allow or deny access to a specific site URL.

See Section 4.2 for further information about the URL request filter.

To configure rules for the URL request filter:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Request Filters**.

   The Request Filters Summary page displays.
3. From the **Site** list, select the site to apply the filter. See Section 2.11.3 and Section 2.11.4 to create additional sites.

   You can configure filters and filter rules for specific sites or **Undefined Sites**. Oracle Web Cache directs client requests that do not match a defined site to the request filters configured for **Undefined Sites**.
4. Click the **URL** link.

   The URL Request Filter page displays.
5. From the **Audit** list, select the level of action for Oracle Web Cache to include in the audit log for the request filter.
6. From the **Response to deny** list, select the HTTP response for Oracle Web Cache to return to browsers for requests that are denied by this request filter.

   The **Close Connection** option does not return any HTTP responses. It just closes the connection.
7. In the **Defined Rules** section, create a new rule:
   a. Click **Create** to create a row in the table.
   b. Click the **Enable** check box to enable the rule; deselect the check box to disable the rule temporarily without losing the rule definition.
   c. In the **URL** field, based on the **URL Type** you select, enter an optional URL string. If no URLs are specified, then all requests are checked. It is equivalent to specifying a URL with a prefix `/`.

      - **Path Prefix:** Enter the path prefix of the objects. Start the path with `/`; do not start the path with `http://host_name:port/`. 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 (\).

      - **File Extension:** Enter the file extension. Because Oracle Web Cache internally starts the file extension with a period (.), it is not necessary to enter it.

      - **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.
   d. From the **URL Type** list, select an option to determine how the rule's URL are compared to the request's URL:

      - **Path Prefix:** Select to allow or deny access to requests matching a path prefix.

      - **File Extension:** Select to allow or deny access to requests matching a particular file extension.

      - **Regular Expression:** Select to allow or deny access to requests matching regular expression syntax.

e. Click the **Case Insensitive Match** check box to match requests regardless of the case. If you do not select this check box, the rule bases the match on the case.

f. Click the **Allow** check box for Oracle Web Cache to allow requests matching the URL fields; deselect the check box for Oracle Web Cache to deny requests matching the IP address and URL fields.

g. Click the **Monitor Only** check box to see the match activity of the rule without enforcing the rule.

When results are expected, then disable **Monitor Only** to enforce the rule. See Section 4.4 for further information about the **Monitor Only** option.

h. Click **Apply** to save the rule settings.

8. Perform Step 7 for any additional rules. You can also add learned rules, as described in Section 4.8.1.

9. Modify the **Catch All** rule, keeping in mind it is applied to all requests that do not match a defined rule.

Oracle recommends creating allow rules, followed by a **Catch All** deny rule.

10. Use the **Move Up** and **Move Down** icons to change the order in which the rules are matched against requests.

The order of the rules is important. Oracle Web Cache matches higher priority rules first.

11. Click the **Request Filters Summary** breadcrumb at the top of the page, or from the **Web Cache** menu, select **Administration** and then **Request Filters** to navigate back to the Request Filters Summary page.

12. In the **URL** row, click **Enable** to enable the filter.

If you do not click **Enable**, Oracle Web Cache ignores any configured filter rules for this filter.

13. Click **Apply** to save the configuration for the request filter.

4.8.1 Activating Learned Rules for the URL Request Filter

See Section 4.3 for further information about how learned rules are collected from the **Catch All** rule. You can add learned rules to the URL request filter.

To enable learned rules for the URL request filter:

1. Navigate to the URL Request Filter page. See Section 2.7.2.

2. In the **Learned Rules** section, if you see that some learned rules have been suggested, monitor the statistics for these rules by watching this page for awhile. When you decide that one or more learned rules make sense, proceed.

If no learned rules display under **Catch All Rule**, then there are no learned rules.

3. Select the row, and click **Add Rule to Defined Rules** to activate the rule and move it the **Defined Rules** section.

4. Use the **Move Up** and **Move Down** icons to change the order in which the rules are matched against requests.

5. Click **Apply** to save the configuration for the request filter.

4.9 Configuring Rules for the Header Request Filter

The header request filter enables Oracle Web Cache to deny access based on HTTP header values. Rules for the header request filter are most effective for white box lists.

See Section 4.2 for further information about the header request filter.

To configure rules for the header request filter:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Request Filters**.

   The Request Filters Summary page displays.
3. From the **Site** list, select the site to apply the filter. See Section 2.11.3 and Section 2.11.4 to create additional sites.

   You can configure filters and filter rules for specific sites or **Undefined Sites**. Oracle Web Cache directs client requests that do not match a defined site to the request filters configured for **Undefined Sites**.
4. Click the **Header** link.

   The Header Request Filter page displays.
5. From the **Audit** list, select the level of action for Oracle Web Cache to include in the audit log for the request filter.
6. From the **Response to deny** list, select the HTTP response for Oracle Web Cache to return to browsers for requests that are denied by this request filter.

   The **Close Connection** option does not return any HTTP responses. It just closes the connection.
7. Create a new rule:
   a. Click **Create** to create a row in the table.
   b. In the **Header Name** field, enter the name of the HTTP request header name, such as `Cookie`.
   c. In the **Value Expression** field, enter the expression, as a substring or regular expression, for the header's value used to compare against an incoming request.
   d. From the **Type** list, select to base the match evaluation on the substring or regular expression in the **Value Expression** field.
   e. Click the **Enable** check box to enable the rule; deselect the check box to disable the rule temporarily without losing the rule definition.
   f. In the **URL** field, based on the **URL Type** you select, enter an optional URL string. If no URLs are specified, then all requests are checked. It is equivalent to specifying a URL with a prefix `/`.

      - **Path Prefix:** Enter the path prefix of the objects. Start the path with `/`; do not start the path with `http://`*`host_name`*`:`*`port`*`/`. 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 (\).

      - **File Extension:** Enter the file extension. Because Oracle Web Cache internally starts the file extension with a period (.), it is not necessary to enter it.

- **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.

g. From the **URL Type** list, select an option to determine how the rule's URL is compared to the request's URL:

- **Path Prefix:** Select to allow or deny access to requests matching a path prefix.

- **File Extension:** Select to allow or deny access to requests matching a particular file extension.

- **Regular Expression:** Select to allow or deny access to requests matching regular expression syntax.

h. Click the **Case Insensitive Match** check box to match requests regardless of the case. If you do not select this check box, the rule bases the match on the case.

i. Click the **Match If Found** check box for Oracle Web Cache to match incoming requests in which the header value matches the substring or regular expression specified in the **Value Expression** field. If there is a match with a request and the rule is enabled, the filter denies the request.

Do not select the **Match If Found** check box for Oracle Web Cache to match incoming requests in which the header value does not match the substring or regular expression specified in the **Value Expression** field. Oracle Web Cache denies the request only if the string or expression is not found, meaning that the request is allowed if the string is found.

Create a rule with the **Match If Found** check box selected, followed by rules without the check box selected.

j. Click the **Monitor Only** check box to see the match activity of the rule without enforcing the rule.

When results are expected, then disable **Monitor Only** to enforce the rule. See Section 4.4 for further information about the **Monitor Only** option.

k. Click **Apply** to save the rule settings.

8. Perform Step 7 for any additional rules.

9. Use the **Move Up** and **Move Down** icons to change the order in which the rules are matched against requests.

The order of the rules is important. Oracle Web Cache matches higher priority rules first.

10. Click the **Request Filters Summary** breadcrumb at the top of the page, or from the **Web Cache** menu, select **Administration** and then **Request Filters** to navigate back to the Request Filters Summary page.

11. In the **Header** row, click **Enable** to enable the filter.

If you do not click **Enable**, Oracle Web Cache ignores any configured filter rules for this filter.

12. Click **Apply** to save the configuration for the request filter.

4.10 Configuring Rules for the Query String Request Filter

The query string request filter enables Oracle Web Cache to deny access based on query string parameter values.

See Section 4.2 for further information about the query-string request filter.

To configure rules for the query string request filter:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Request Filters**.

   The Request Filters Summary page displays.
3. From the **Site** list, select the site to apply the filter. See Section 2.11.3 and Section 2.11.4 to create additional sites.

   You can configure filters and filter rules for specific sites or **Undefined Sites**. Oracle Web Cache directs client requests that do not match a defined site to the request filters configured for **Undefined Sites**.
4. Click the **Query String** link.

   The Query String Request Filter page displays.
5. From the **Audit** list, select the level of action for Oracle Web Cache to include in its audit log for the request filter.
6. From the **Response to deny** list, select the HTTP response for Oracle Web Cache to return to browsers for requests that are denied by this request filter.

   The **Close Connection** option does not return any HTTP responses. It just closes the connection.
7. Create a new rule:
   a. Click **Create** to create a row in the table.
   b. In the **Query String Expression** field, enter the query string, as a substring or regular expression, to compare against an incoming request.
   c. Click the **Enable** check box to enable the rule; deselect the check box to disable the rule temporarily without losing the rule definition.
   d. From the **Type** list, select to base the match evaluation on the substring or regular expression in the **Query String Expression** field.
   e. Click the **Match If Found** check box for Oracle Web Cache to match incoming requests in which the query string matches the substring or regular expression specified in the **Value Expression** field. If there is a match with a request and the rule is enabled, the filter denies the request.

      Do not select the **Match If Found** check box for Oracle Web Cache to match incoming requests in which the query string does not match the substring or regular expression specified in the **Value Expression** field. Oracle Web Cache denies the request only if the string or expression is not found, meaning that the request is allowed if the string is found.

      For example, if you specify a rule with a **Query String Expression** of `abc`, **Type** of **substring**, and do not select the **Match If Found** check box, the filter would deny a request which did not contain the string `abc` in the query string (or POST body). It would allow a request which contains the string `abc`.

      You can create multiple rules to allow requests with a certain string and deny requests with another string. For example, if you specify a second rule with a **Query String Expression** of `def`, **Type** of **substring**, and click the **Match If Found** check box, the filter would allow a request with `abc` in the query string but would deny a request with `def` in the query string.

f. In the **URL** field, based on the **URL Type** you select, enter an optional URL string. If no URLs are specified, then all requests are checked. It is equivalent to specifying a URL with a prefix `/`.

   - **Path Prefix:** Enter the path prefix of the objects. Start the path with `/`; do not start the path with `http://host_name:port/`. 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 (\).

   - **File Extension:** Enter the file extension. Because Oracle Web Cache internally starts the file extension with a period (.), it is not necessary to enter it.

   - **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.

g. From the **URL Type** list, select an option to determine how the rule's URL are compared to the request's URL:

   - **Path Prefix:** Select to allow or deny access to requests matching a path prefix.

   - **File Extension:** Select to allow or deny access to requests matching a particular file extension.

   - **Regular Expression:** Select to allow or deny access to requests matching regular expression syntax.

h. Click the **Case Insensitive Match** check box to match requests regardless of the case. If you do not select this check box, the rule bases the match on the case.

i. Click the **Monitor Only** check box to see the match activity of the rule without enforcing the rule.

   When results are expected, then disable **Monitor Only** to enforce the rule. See Section 4.4 for further information about the **Monitor Only** option.

j. Click **Apply** to save the rule settings.

8. Perform Step 7 for any additional rules.

9. Use the **Move Up** and **Move Down** icons to change the order in which the rules are matched against requests.

   The order of the rules is important. Oracle Web Cache matches higher priority rules first.

10. Click the **Request Filters Summary** breadcrumb at the top of the page, or from the **Web Cache** menu, select **Administration** and then **Request Filters** to navigate back to the Request Filters Summary page.

11. In the **Query String** row, click **Enable** to enable the filter.

    If you do not click **Enable**, Oracle Web Cache ignores any configured filter rules for this filter.

12. Click **Apply** to save the configuration for the request filter.

4.11 Configuring Rules for the Format Request Filter

The format request filter enables Oracle Web Cache to deny access based on well-formed and valid URLs.

See Section 4.2 for further information about the format request filter.

To configure rules for the format request filter:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Request Filters**.

   The Request Filters Summary page displays.
3. From the **Site** list, select the site to apply the filter. See Section 2.11.3 and Section 2.11.4 to create additional sites.

   You can configure filters and filter rules for specific sites or **Undefined Sites**. Oracle Web Cache directs client requests that do not match a defined site to the request filters configured for **Undefined Sites**.
4. Click the **Query String** link.

   The Query String Request Filter page displays.
5. From the **Audit** list, select the level of action for Oracle Web Cache to include in its audit log for the request filter.
6. From the **Response to deny** list, select the HTTP response for Oracle Web Cache to return to browsers for requests that are denied by this request filter.

   The **Close Connection** option does not return any HTTP responses. It just closes the connection.
7. Define a rule for the validation type:
   a. Click the **Enable** check box to enable the validation check:

      - **Null Byte:** This validation checks for encoding as a null byte as `%00`. Most applications do not expect null bytes in the URL. This may cause a string which contains tricks after a null byte to pass an application check because the application stops checking when it hits the null byte, thinking that it is the end of string marker.

      - **Valid Unicode:** This validation checks for Unicode characters, either encoded or raw in the URL for an application that is not set up to handle Unicode.

      - Strict Encoding: This validation checks for unencoded characters, such as a space, backslash (\), or non-printable characters.

      - **Double Encoding:** This validation checks for `%XY` sequences using `%XY` encoding, in an attempt to get the `%XY` sequence to be passed to the application. This could allow the hacker to specify a character that would otherwise be rejected.

      - **Uencoded Unicode Characters:** This validation checks for Unicode characters, either encoded or raw in the URL for an application that is not set up to handle Unicode.

   b. Click the **Check Query String** check box to verify the format of the URL, as well as the query string or request body for a POST request; leave this option unchecked to verify only the format of the URL.
   c. Click the **Allow** check box for Oracle Web Cache to allow requests containing the invalid format; deselect the check box for Oracle Web Cache to deny requests containing the invalid format.
   d. Click the **Monitor Only** check box to see the match activity of the rule without enforcing the rule.

When results are expected, then disable **Monitor Only** to enforce the rule. See Section 4.4 for further information about the **Monitor Only** option.

e. Click **Apply** to save the rule settings.

8. Click the **Request Filters Summary** breadcrumb at the top of the page, or from the **Web Cache** menu, select **Administration** and then **Request Filters** to navigate back to the Request Filters Summary page.
9. In the **Format** row, click **Enable** to enable the filter.

   If you do not click **Enable**, Oracle Web Cache ignores any configured filter rules for this filter.
10. Click **Apply** to save the configuration for the request filter.

4.12 Deleting Rules for a Request Filter

To delete a rule for a request filter:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Request Filters**.

   The Request Filters Summary page displays.
3. Click the filter you want to modify.
4. From the **Site** list, select the site.
5. Select a rule in the table and click the **Delete** icon.
6. Repeat Step 5 for each additional rule you want to remove.
7. Click **Apply** to save the configuration for the request filter.

4.13 Monitoring Statistics for Request Filter Types and Rules

Fusion Middleware Control provides statistics for assessing the effectiveness of configured request filters and rules. By analyzing the rules, you can determine if you prioritized the rules incorrectly. For example, if the you notice a deny rule is matched but configured allow rules are never matched, then prioritize the allow rules first.

If you make changes to the configuration settings for Oracle Web Cache, Oracle Web Cache disables the request-filter statistics and labels them as **NA**.

To view request-filter statistics:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Request Filters**.

   The Request Filters Summary page displays.
3. Scroll to the far right, to view the **Request Statistics** and **Post Allow Statistics**. The **Request Statistics** display the run time statistics for tracking how the configured request filters handle the incoming requests:
   - **Matched Catch All:** Displays the number of requests that matched the Catch All rules.
   - **Matched Others:** Displays the number of requests that matched the rules that were not Catch All requests.

- **Matched Denied:** Displays the number of requests that matched the rule and were denied by Oracle Web Cache.

If there are few requests matching the filter, then consider changing the rules for the filter to improve its effectiveness.

The **Post Allow Statistics** display the statistics for allowed requests:

- **Succeeded:** Displays the number of requests that were allowed and succeeded.
- **Denied Later:** Displays the number of requests that were allowed, but subsequently denied by another filter.
- **Failed:** Displays the number of requests that were allowed, but the request failed with an error.

4. To gather more information for a particular filter, click the filter to view the individual rules and accompanying statistics.

4.14 Reducing Time to Configure Request Filters

This section covers the following configuration tasks for easing configuration work. These features are only available in Oracle Web Cache Manager.

- Section 4.14.1, "Copying Rules from a Source Site to a Target Site"
- Section 4.14.2, "Reverting Configuration Settings"

4.14.1 Copying Rules from a Source Site to a Target Site

You can reduce the time spent configuring filters and associated rules by completing the configuration for one site and applying the configuration to other sites. You can copy the complete configuration for all filters, or you can copy the configuration for the rules for a specific filter.

To copy the complete configuration for all the filters from a source site to a target site:

1. Configure settings for the various filters, as described in Section 4.5 to Section 4.11.
2. From Oracle Web Cache Manager, in the navigator frame, select **Filtering** > **Request Filters**. See Section 2.7.2.

   The Request Filters Summary page displays.
3. From the **For Site** list, select the source site with the complete configuration you want to copy.
4. Click **Copy All Filter**s toward the bottom of the page.
5. In the Copy All Request Filter dialog, from the **To Site** list, select the target site to apply the configuration settings, and then click **Submit**.
6. Click **Apply Changes**.

To copy the rule configuration for a specific filter from a source site to a target site:

1. Configure settings for the various filters, as described in Section 4.5 to Section 4.11.
2. From Oracle Web Cache Manager, in the navigator frame, select **Filtering** > **Request Filters**. See Section 2.7.2.

   The Request Filters Summary page displays.
3. Select a specific filter from the **Filter Type** column.

The configuration page for the selected filter displays.

4. From the **For Site** list, select the source site with the complete configuration you want to copy.
5. Click **Copy Filter** toward the bottom of the page.
6. In the Copy Request Filter dialog, from the **To Site** list, select the target site to apply the configuration settings, and then click **Submit**.
7. Click **Apply Changes**.

4.14.2 Reverting Configuration Settings

You can revert to the original configuration settings provided by Oracle Web Cache for all filters or a specific filter.

To revert the configuration settings for all filters:

1. From Oracle Web Cache Manager, in the navigator frame, select **Filtering** > **Request Filters**. See Section 2.7.2.

   The Request Filters Summary page displays.
2. From the **For Site** list, select the site you want to revert configuration settings.
3. Click **Clear All Filters**.
4. Click **Apply Changes**.

To revert the configuration settings for a specific filter:

1. From Oracle Web Cache Manager, in the navigator frame, select **Filtering** > **Request Filters**. See Section 2.7.2.

   The Request Filters Summary page displays.
2. Select a specific filter from the **Filter Type** column.

   The configuration page for the selected filter displays.
3. From the **For Site** list, select the site you want to revert configuration settings.
4. Click **Clear Filter**.
5. Click **Apply Changes**.

5 Configuring Security

The ability to control user access to Web content and to protect against intrusion is the critical issue affecting enterprise deployment. This chapter describes how to configure security for Oracle Web Cache.

For general information about security, see the *Oracle Fusion Middleware Security Guide*.

This chapter includes the following topics:

5.1 Introduction to Security in Oracle Web Cache

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

5.1.1 Oracle Web Cache Security Model

Oracle Web Cache provides the following security-related features:

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

5.1.1.1 Restricted Administration

Oracle Web Cache restricts administration with the following features:

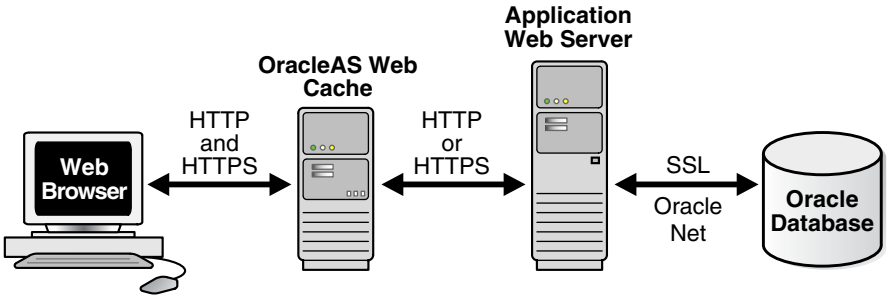
- Password authentication for administration and invalidation operations
- Control over which ports are used for administration and invalidation operations
- IP and subnet administration restrictions

5.1.1.2 Secure Sockets Layer (SSL)

The **HTTPS protocol** (HTTP over SSL) is used to encrypt network traffic. Oracle Web Cache supports HTTPS for all of its network traffic, including HTTP clients, administration, invalidation, and statistics requests, and to communicate with its origin servers and **cache cluster** peers.

As shown in Figure 5–1, you can configure Oracle Web Cache to receive HTTPS client requests and send HTTPS requests to origin servers.

Figure 5–1 SSL for Secure Connections



When sending requests to origin servers, note that HTTPS traffic can be processor intensive. If traffic from Oracle Web Cache to an origin server must travel over the open Internet, configure Oracle Web Cache to send HTTPS requests to the origin servers. If traffic only travels through a LAN in a data center, then consider using HTTP to reduce load on the origin servers.

Oracle Web Cache supports both server-side and client-side certificates. SSL server certificates can be used to verify the authenticity of the server, and SSL client certificates can be used to restrict access to certain clients. SSL however is generally not used alone for user verification.

This section interacts with the following entities:

- Section 5.1.1.2.1, "Certificate Authority"
- Section 5.1.1.2.2, "Certificate"
- Section 5.1.1.2.3, "Wallet"

5.1.1.2.1 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 Oracle 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.

5.1.1.2.2 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.

Oracle 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, Oracle 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).

  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.

- **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. Certificate Revocation Lists (CRLs) validate the peer certificate in the SSL handshake and ensure that the certificate is not on the list of revoked certificates issued by the 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, Oracle Web Cache adds HTTP headers to the request. These headers begin with the string `SSL-Client-Cert`.

  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 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 the listening port requires client certificates, then the connection is refused. If the site requires client certificates and the SSL port does not, then HTTP error 403 Forbidden returns.
- Oracle Web Cache supports the use of client-side certificates with Oracle HTTP Server only.
- Oracle Web Cache does not support client-side certificates with a distributed cache hierarchy because the security of the certificates cannot be guaranteed.
- Although the Oracle HTTP Server supports OpenSSL certificate revocation lists, Oracle 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 Fusion Middleware 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 Fusion Middleware.

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

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

To configure HTTPS for Oracle Web Cache, create a wallet on the Oracle Web Cache server for each supported site. You specify the location of the wallet for each of the Oracle 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 Oracle 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 *Oracle Fusion Middleware Administrator's Guide* for further information about Oracle Wallet Manager.

5.1.1.2.4 How SSL Works To describe how SSL works in an HTTPS connection, the word client is used to describe either a browser or Oracle Web Cache, and the word server is used to describe either Oracle Web Cache or an origin server. For example, when a browser is the client, the server can be Oracle Web Cache or an origin server; when Oracle 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 and server determine which cipher (encryption algorithm) to use.
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 cipher.

5.1.1.3 SSL Acceleration

Oracle Web Cache provides SSL acceleration by moving the SSL processing to the Web tier.

In addition to off-board SSL acceleration solutions, Oracle Fusion Middleware supports both software-only SSL operations and nCipher's BHAPI-compliant hardware for deployment on servers running Oracle 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 off loads 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 `http://www.ncipher.com` for more information about nCipher.

5.1.2 Resources Protected

By default, the user that performed the installation is the owner of Oracle Web Cache files. Most files are readable by the user ID specified in the Process Identity page of Oracle Web Cache Manager (**Properties** > **Process Identity**).

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

5.1.3 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. Oracle Web Cache does not restrict IP address restrictions on a URL basis. If you are using `mod_access` with Oracle 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. For more information, see the *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*.

5.1.4 Leveraging Oracle Identity Management Infrastructure

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

- Section 5.1.4.1, "About Caching Content from Oracle Single Sign-On Servers"
- Section 5.1.4.2, "About Caching Oracle Single Sign-On Partner Applications (mod_osso)"
- Section 5.1.4.3, "About Authentication through Oracle Single Sign-On"

5.1.4.1 About Caching Content from Oracle Single Sign-On Servers

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

5.1.4.2 About Caching Oracle Single Sign-On Partner Applications (mod_osso)

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

5.1.4.3 About Authentication through Oracle Single Sign-On

You can configure Oracle Web Cache to require authentication through Oracle Single Sign-On. Incoming requests must have a valid Oracle Single Sign-On cookie to be served by Oracle Web Cache. See Section 5.8 for configuration details.

5.2 Configuring Password Security

Before submitting invalidation and statistics monitoring requests, establish secure passwords for sending the requests.

The `invalidator` account is an administrator authorized to send invalidation requests. The `invalidator` account sends HTTP POST requests to invalidate objects in the cache.

The `administrator` account is the Oracle Web Cache administrator authorized to log in to Oracle Web Cache Manager and make configuration changes through that interface. This administrator is also authorized to send statistic monitoring requests to the Oracle Web Cache statistics monitoring port. If after monitoring metrics from Fusion Middleware Control you need additional performance metrics, you can access the statistic monitoring port with the `administrator` account to view detailed performance metrics. See Section 8.4.

The default password for these accounts is the password you supplied in the Web Cache Administrator page of the Oracle Universal Installer. Before you begin configuration, change the passwords for these accounts to a secure password. You must perform this configuration in Fusion Middleware Control.

To establish secure passwords for the `invalidator` and `monitor` accounts:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** > **Passwords**.

   The Passwords page displays.
3. In the **New Password** field, enter a new password, keeping the following restrictions in mind:
   - Passwords must be between 5 and 30 characters.
   - At least one character 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.
4. In the **Confirm Password** field, reenter the new password to confirm you entered the password correctly.
5. Click **Apply.**
6. Restart Oracle Web Cache. See Section 2.13.

   If Oracle Web Cache is not restarted, you may encounter an error when accessing some Fusion Middleware Control pages.

5.3 Configuring Access Control

By default, the computer on which you installed Oracle Web Cache is the trusted host. You can change the trusted subnet or trusted host from which administration, invalidation, and statistics monitoring requests can take place.

To specify if some or all of the traffic to an application is restricted to use HTTPS:

1. From Oracle Web Cache Manager, in the navigator frame, select **Properties** > **Security**. See Section 2.7.2.

   The Security page appears.

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

   The Change Trusted Subnets dialog box appears.

3. Select an option:

   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 using these format:

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

     Example: 10.1.0.0

   – Network/netmask pair for subnet restriction through masking

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

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

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

4. Click **Submit.**

5. Restart Oracle Web Cache using `opmnctl`. See Section 2.13.1.

5.4 Configuring Oracle Web Cache for HTTPS Requests

To provide more security for your Web site, you can configure Oracle Web Cache to receive HTTPS protocol client requests and send HTTPS requests to the origin server. HTTPS uses SSL to encrypt and decrypt user page requests as well as the pages that are returned by the Oracle Web Cache and origin servers. You can also configure Oracle Web Cache to send traffic to the origin server through an HTTPS listening port.

To configure HTTPS support for Oracle Web Cache, perform these tasks:

- Section 5.4.1, "Task 1: Create Wallets"
- Section 5.4.2, "Task 2: Configure an HTTPS Listening Port"
- Section 5.4.3, "Task 3: Configure SSL Settings for Oracle Web Cache Connections to Origin Servers"
- Section 5.4.4, "Task 4: Configure a Site to Require HTTPS Requests"
- Section 5.4.5, "Task 5: Restart Oracle Web Cache"
- Section 5.4.5, "Task 5: Restart Oracle Web Cache"
- Section 5.4.6, "Task 6: Perform Additional Configuration for Oracle WebLogic Servers"

5.4.1 Task 1: Create Wallets

To support HTTPS for Oracle Web Cache, you must create a **wallet** on the Oracle Web Cache server for each supported site. You need wallets to support the following HTTPS requests:

- Client requests for sites hosted by Oracle Web Cache
- Administration, invalidation, and statistics monitoring requests to Oracle Web Cache
- Oracle Web Cache requests to origin servers, as well as `admin` server process requests for requests to invalidation and statistics monitoring ports enabled for SSL

For each site that Oracle Web Cache supports, configure at least one wallet. You specify the location of the wallet for each of the Oracle 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.

To create a wallet:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Security** and then **Wallets**.

   The Wallets page displays.
3. Perform the tasks in section "Create a Wallet" of the *Oracle Fusion Middleware Administrator's Guide*.

5.4.2 Task 2: Configure an HTTPS Listening Port

To configure HTTPS protocol support between client and Oracle Web Cache, you must configure an HTTPS listening port for Oracle Web Cache.

To add an HTTPS listening port:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. Create the listening port:
   a. From the **Web Cache** menu, select **Administration** > **Ports Configuration**.

      The Ports Configuration page displays.

   b. Click **Create**.

      The Create Port page appears.

   c. From the **Port Type** list, select **NORM**.

   d. In the **IP Address** field, specify the computer running Oracle Web Cache:

      - IP version 4 address written in a 32-bit dotted decimal notation or an IP version 6 address written in a 128-bit notation. See Section 2.5.

      - A host name that resolves to an IP address of the computer running Oracle 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

e. In the **Port** field, enter the listening port from which Oracle Web Cache receives client requests for the Web site.

   Ensure that this port number is not in use.

   Port numbers less than 1024 are reserved for use by privileged processes on UNIX. To configure Oracle Web Cache to listen on a port less than 1024, such as on port 80, run the Oracle Web Cache `webcached` executable with the root privilege. If the `webcached` executable is not run as root, Oracle Web Cache fails to start.

   See Section 5.9 for instructions on changing the `webcached` executable to run as root.

f. Click **OK**.

3. Enable the port for SSL:

   a. From the **Web Cache** menu, select **Security** > **SSL Configuration**.

      The SSL Configuration page displays.

   b. Select the row for the endpoint you created in Step 2 and click **Edit**.

      The Edit Port page displays.

   c. In the **SSL Configuration** section, click **Enable SSL**.

   d. In the **Server Wallet Name** field, select the wallet you created in Section 5.4.1.

   e. In the **Advanced SSL Settings** section, click **Expand** (+) to expand the configuration settings:

   f. From the **Client Authentication** list, select the type of client authentication.

      - **Server Authentication:** A server authenticates itself to a client.

      - **Mutual Authentication:** A client authenticates itself to a server and that server authenticates itself to the client.

      - **No Authentication:** Neither server nor client are required to authenticate.

      - **Optional Client Authentication:** The server authenticates itself to the client, but the client may or may not authenticate itself to the server. Even if the client does not authenticate itself, the SSL session still goes through.

   g. From the **SSL protocol version** list, select the version of SSL to use.

      - **All:** This selection enables the **v1**, **v3**, and **v3-v2Hello** options.

      - **v1:** This selection supports TLS version 1 traffic.

      - **v3:** This selection provides SSL version 3 traffic.

      - **v3_v2Hello:** This selection combines the SSL version 2 hello message format with SSL version 3 handling to support SSL version upgrade during handshake operations.

   h. Click **OK**.

5.4.3 Task 3: Configure SSL Settings for Oracle Web Cache Connections to Origin Servers

In this task, specify which SSL wallet to use for Oracle Web Cache connections to origin servers. This wallet must contain a certificate that matches the wallet used by the origin servers.

To specify which SSL wallet to use for Web Cache connections to origin servers:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Security** and then **SSL Configuration**.

   The SSL Configuration page displays.
3. Click the **Expand** icon next to the **SSL Communication Between Web Cache and Oracle HTTP Server** section.
4. Click **Change Wallet** to display the Select Client Wallet dialog.
5. Select the wallet to use, and click **OK**. Ensure this wallet contains a certificate that matches the wallet used by the origin server.

5.4.4 Task 4: Configure a Site to Require HTTPS Requests

If your environment has a mix of HTTP and HTTPS traffic, follow these instructions to restrict traffic for a specific site (or URL prefix subset of the site), so that the requests must be received by Oracle Web Cache over SSL connections only.

To configure the site settings, use a combination of Fusion Middleware Control and Oracle Web Cache Manager:

1. In the Fusion Middleware Control, specify a site definition and site-to-server mapping, as described in Section 2.11.3 and Section 2.11.4. When configuring the site definition, ensure you specify an HTTPS listening port. This site uses the wallet defined for that port.
2. From the Web Cache menu, select **Availability** > **Restart** to save the configuration settings and restart Oracle Web Cache.
3. From Oracle Web Cache Manager, in the navigator frame, select **Properties** > **Site Definition**. See Section 2.7.2.
4. Select the site you created in Step 1, and click **Show/Edit Site**.
5. In the Show/Edit dialog, in the **HTTPS Only Prefix** field, enter the URL prefix for which only HTTPS requests are served. If all traffic must be restricted to HTTPS, enter "/ " for the entire site.
6. Click **Submit**.

5.4.4.1 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 Oracle 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 Section 2.11.5 for further information about configuring the keep-alive timeout in Oracle Web Cache.

5.4.5 Task 5: Restart Oracle Web Cache

See Section 2.13.

5.4.6 Task 6: Perform Additional Configuration for Oracle WebLogic Servers

If the orgin server is an Oracle WebLogic Server, you need to specify an extra attribute for Oracle Web Cache to process SSL requests correctly:

To configure Oracle Web Cache for a configuration in which the origin server is an Oracle WebLogic Server:

1. Use a text editor to open `webcache.xml`, located in:

   (UNIX) *ORACLE_INSTANCE*/<*instance_name*>/config/WebCache/<*webcache_name*>
   (Windows) *ORACLE_INSTANCE*\<*instance_name*>\config\WebCache\<*webcache_name*>

2. Locate the `HOST ID` element.
3. Add the `SERVERTYPE` attribute to the `CACHE` element. For example:

   ```
   <HOST ID="host_ID" NAME="WLS_server_name" PORT="WLS_server_port"
   LOADLIMIT="100" OSSTATE="ON" SERVERTYPE="WebLogic"/>
   ...
   ```

4. Save `webcache.xml`.
5. Restart Oracle Web Cache with the following command:

   ```
   opmnctl restartproc ias-component=component_name
   ```

   This executable is found in the following directory:

   (UNIX) *ORACLE_INSTANCE*/bin
   (Windows) *ORACLE_INSTANCE*\bin

5.5 Additional HTTPS Configuration

After performing the tasks in Section 5.4, you can perform the following optional configuration:

- Section 5.5.1, "Configuring HTTPS Operation Ports"
- Section 5.5.2, "Requiring Client-Side Certificates"
- Section 5.5.3, "Configuring Certificate Revocation Lists (CRLs)"

5.5.1 Configuring HTTPS Operation Ports

To configure HTTPS ports to listen for administration, invalidation, or statistics monitoring requests in Fusion Middleware Control:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. Create the listening port:
   a. From the **Web Cache** menu, select **Administration** > **Ports Configuration**.

The Ports Configuration page displays.

b. Click **Create**.

The Create Port page appears.

c. From the **Port Type** list, select the port type, **ADMINISTRATION**, **INVALIDATION**, or **STATISTICS**.

d. In the **IP Address** field, specify the computer running Oracle Web Cache:

- IP version 4 address written in a 32-bit dotted decimal notation or an IP version 6 address written in a 128-bit notation. See Section 2.5.

- A host name that resolves to an IP address of the computer running Oracle 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

e. In the **Port** field, enter the listening port from which Oracle Web Cache receives client requests for the Web site.

Ensure that this port number is not in use.

Port numbers less than 1024 are reserved for use by privileged processes on UNIX. To configure Oracle Web Cache to listen on a port less than 1024, such as on port 80, run the Oracle Web Cache `webcached` executable with the root privilege. If the `webcached` executable is not run as root, Oracle Web Cache fails to start.

See Section 5.9 for instructions on changing the `webcached` executable to run as root.

f. Click **OK**.

3. Enable the port for SSL:

a. From the **Web Cache** menu, select **Security** > **SSL Configuration**.

The SSL Configuration page displays.

b. Select the row for the endpoint you created in Step 2 and click **Edit**.

The Edit Port page displays.

c. In the **SSL Configuration** section, click **Enable SSL**.

d. In the **Server Wallet Name** field, select the wallet you created in Section 5.4.1.

e. In the **Advanced SSL Settings** section, click **Expand** (+) to expand the configuration settings:

f. From the **SSL Authentication** list, select the type of client authentication.

- **Server Authentication:** A server authenticates itself to a client.

- **Mutual Authentication:** A client authenticates itself to a server and that server authenticates itself to the client.

- **No Authentication:** Neither server nor client are required to authenticate.

- **Optional Client Authentication:** The server authenticates itself to the client, but the client may or may not authenticate itself to the server. Even if the client does not authenticate itself, the SSL session still goes through.

g. From the **SSL Protocol Version** list, select the version of SSL to use.

h. Click **OK**.

5.5.2 Requiring 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, Oracle 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, Oracle 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.
- If a site requires client certificates, then a `403 Forbidden` error returns if a client certificate is not provided. If a listen port requires client certificates, then the SSL handshake fails if a client certificate is not provided.

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

Oracle 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:

- Section 5.5.2.1, "Configuring Client-Side Certificate Settings for the HTTPS Listening Ports"
- Section 5.5.2.2, "Configuring Client-Side Certificate Settings for Cache Clusters"
- Section 5.5.2.3, "Configuring Client-Side Certificate Settings for a Site"

5.5.2.1 Configuring Client-Side Certificate Settings for the HTTPS Listening Ports

To use client-side certificates, you must enable an HTTPS listening port, as described in Section 5.4.2. If you have a cache cluster, you must enable HTTPS listening ports for all cluster members. In addition, you must configure Oracle Web Cache to require client browsers to provide SSL certificates.

After configuring the client-side certificate, to enable Oracle Web Cache to transfer certificate information to Oracle HTTP Server, add the `AddCertHeader` directive to `httpd.conf`. See the *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server* for information about adding the `AddCertHeader` directive.

5.5.2.2 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 Oracle Web Cache Manager:

1. From Oracle Web Cache Manager, in the navigator frame, select **Properties** > **Security**. See Section 2.7.2.
2. In the **Security Header Configuration** section of the Security page, set the value of **Accept SSL client certificates encoded in SSL-Client-Cert HTTP headers** to **NO** (the default), so Oracle Web Cache does not accept the certificate information in HTTP headers. This setting prevents caches in a cache cluster from accepting the certificate information in HTTP headers.
3. In the **Cluster Security Configuration** section, set the value of the **Route requests that contain SSL client certificates to cache cluster peers** to **YES**, enabling Oracle Web Cache to pass information about the client-side certificate in HTTP headers to a peer cache. This setting is used for caches in a cache cluster so that they can pass the information to a peer cache.
4. Click **Apply Changes**.
5. Restart Oracle Web Cache. See Section 2.13.

5.5.2.3 Configuring Client-Side Certificate Settings for a Site

You can also specify that an entire site require client-side certificates. If a site requires client certificates, then a `403 Forbidden` error returns if a client certificate is not provided.

To configure a site to use client-side certificates:

1. From Oracle Web Cache Manager, in the navigator frame, select **Properties** > **Site Definition**. See Section 2.7.2.
2. In the Site Definitions page, select the site and click **Show/Edit Site.**
3. In the Show/Edit dialog, in the **Client-Side Certificate** field, select **Required**.
4. Click **Submit.**
5. Restart Oracle Web Cache. See Section 2.13.

5.5.3 Configuring Certificate Revocation Lists (CRLs)

Fusion Middleware Control or Oracle Web Cache Manager do not provide support for client certificate validation with Certificate Revocation Lists (CRLs). You can configure this support by manually editing the `webcache.xml` file.

Client certificate revocation status is checked against CRLs that are located in a file system directory. Typically, CRL definitions are valid for a few days, and must be updated on a regular basis. Whenever the CRL definitions are modified, you must restart Oracle Web Cache.

When CRL validation is enabled and available, Oracle Web Cache performs certificate revocation status checking for client certificates. The SSL connection is rejected if a certificate is revoked. SSL connections are accepted if no CRL is found, or if the certificate has not been revoked.

To configure certificate validation with CRL

1. Enable client certificate for the HTTPS listen port. See Section 5.5.2.
2. Use a text editor to open `webcache.xml`, located in:

   ```
   (UNIX) ORACLE_INSTANCE/<instance_name>/config/WebCache/<webcache_name>
   (Windows) ORACLE_INSTANCE\<instance_name>\config\WebCache\<webcache_name>
   ```

3. Locate the HTTPS listen port in `webcache.xml` for which CRL checking needs to be enabled, and add the `SSLCRLENABLE="YES"` parameter to the `LISTEN` directive. For example:

   ```
   ...
   <LISTEN IPADDR="ANY" PORT="443" PORTTYPE="NORM" SSLENABLED="SSLV3_V2H"  CLIENT_
   CERT="YES" SSLCRLENABLE="YES" STRONG_CRYPTO_ONLY="NO">
   ...
   ```

4. Configure CRL file location by adding the `SSLCRLPATH` and `SSLCRLFILE` parameters to the HTTPS `LISTEN` directive.

   - `SSLCRLPATH`: Enter the path to the directory where CRLs are stored. Ensure that the path is correct; otherwise CRL checking will not work. This parameter has no default value.
   - `SSLCRLFILE`: Enter the path to a comprehensive CRL file where PEM-encoded (BASE64 CRLs are concatenated in order of preference in one file. If this parameter is set, then the file must be present at the specified location. Otherwise CRL checking will not work.

   For example:

   ```
   ...
   <LISTEN IPADDR="ANY" PORT="443" PORTTYPE="NORM" SSLENABLED="SSLV3_V2H"  CLIENT_
   CERT="YES" SSLCRLENABLE="YES"  SSLCRLFILE="/ORACLE_HOME/webcache/crls/sample_
   crl" SSLCRLPATH="/ORACLE_HOME/webcache/crls/" STRONG_CRYPTO_ONLY="NO">
   ...
   ```

   Use the command line utility `orapki` to rename CRLs in your file system. See section "Certificate Revocation List Management" in the *Oracle Database Advanced Security Administrator's Guide* from the Oracle Database documentation library for information about using `orapki`.

5. Save `webcache.xml`.
6. Restart Oracle Web Cache with the following command:

   ```
   opmnctl restartproc ias-component=component_name
   ```

   This executable is found in the following directory:

   ```
   (UNIX) ORACLE_INSTANCE/bin
   (Windows) ORACLE_INSTANCE\bin
   ```

   In a cluster configuration, when configuration changes are made directly to a cluster member's `webcache.xml` file, use Fusion Middleware Control or Oracle Web Cache Manager to propagate the change to other cluster members. See Section 3.6 and Section 3.7.

5.6 Configuring HTTP Request Header Size

By default, Oracle 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,Oracle 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, Oracle 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:

1. From Oracle Web Cache Manager, in the navigator frame, select **Properties** > **Security**. See Section 2.7.2.

   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**.

6. Restart Oracle Web Cache. See Section 2.13.

5.7 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, Oracle 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, Oracle 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:

1. From Oracle Web Cache Manager, in the navigator frame, select **Properties** > **Security**. See Section 2.7.2.
2. In the 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**, Oracle 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**, Oracle 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 Security Header Configuration dialog:
   - For a simple configuration, the value should be **NO**.
   - In a cache cluster, the value should be **NO** for all cluster members.
   - 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, Oracle 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**.
5. Restart Oracle Web Cache. See Section 2.13.

5.8 Configuring Support for Caching Secured Content

You can configure Oracle Web Cache to support caching content that is secured by Oracle Single Sign-On authentication with no other authorization requirements.

To enable this setting in Oracle Web Cache Manager:

1. From Oracle Web Cache Manager, in the navigator frame, select **Origin Servers, Sites, and Load Balancing** > **Site Definitions.** See Section 2.7.2.
2. Select a configured site and click **Edit Show/Edit Site**.
3. In the For Site dialog, in the **Attributes** section, select the type of authentication required for requested objects:
   - **Oracle Single Sign-On:** Select to require authentication through Oracle Single Sign-On. Oracle Web Cache requires a valid Oracle Single Sign-On cookie to serve requests.
   - **None:** Select to not require any authentication.
4. Click **Submit.**
5. Restart Oracle Web Cache. See Section 2.13.

5.9 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 Oracle Web Cache listening ports.
- There are more than 1,024 file descriptors being used for connections to Oracle Web Cache.
- The current `opmnctl` user does not match the configured process identity user in the Process Identity page (**Properties** > **Process Identity**) of Oracle Web Cache Manager.

This section contains the following topics:

5.9.1 Configuring Process Identity

By default, the user that performed the installation is the owner of Oracle 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 of Oracle Web Cache Manager, the Oracle 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 Oracle Web Cache fails to start.

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

1. From Oracle Web Cache Manager, in the navigator frame, select **Properties** > **Process Identity**. See Section 2.7.2.

   The Process Identity page appears.

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

   The Change Process Identity dialog box appears.

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

4. Click **Submit.**

5. 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:

   - `webcache.xml` configuration file in:

```
(UNIX) ORACLE_INSTANCE/<instance_name>/config/WebCache/<webcache_name>
(Windows) ORACLE_INSTANCE\<instance_name>\config\WebCache\webcache_name>
```

- Event and access log files in:

```
(UNIX) ORACLE_INSTANCE/diagnostics/logs/WebCache/<webcache_name>
(Windows) ORACLE_INSTANCE\diagnostics\logs\WebCache\<webcache_name>
```

6. Restart Oracle Web Cache using `opmnctl`. See Section 2.13.1.

5.9.2 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 Oracle Web Cache, you have two options:

- Raise the limit for the particular user that is running Oracle Web Cache. Oracle recommends this mechanism. Refer to operating-system documentation for further information about raising the limit for a user.
- Use the `setroot` command of `webcache_setuser.sh` to provide Oracle Web Cache with root privilege without requiring changing the process identity settings

Every time you upgrade Oracle Web Cache or apply a patch, the Oracle Web Cache binaries are relinked implicitly. Therefore, you must rerun the `setroot` command, as specified in the following procedure.

To use the `setroot` command of `webcache_setuser.sh`:

1. From `$ORACLE_HOME/webcache/bin`, execute:

   ```
   webcache_setuser.sh setroot user_ID
   ```

   where *`user_ID`* is the user that performed installation. See Section 5.10 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. Restart Oracle Web Cache using `opmnctl`. See Section 2.13.1.

5.9.3 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 Oracle Web Cache processes and use the `setidentity` command of `webcache_setuser.sh` to provide Oracle Web Cache with root privilege:

1. Change the process identity of the Oracle Web Cache processes.

   Oracle recommends running Oracle Web Cache using a restricted user. See Section 5.9.1 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 Oracle 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 Section 5.10 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. Restart Oracle Web Cache using `opmnctl`. See Section 2.13.1.

5.9.4 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 cannot 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 Section 5.10 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. Restart Oracle Web Cache using `opmnctl`. See Section 2.13.1.

5.10 Script for Setting File Permissions on UNIX

For UNIX operating systems, use the `webcache_setuser.sh` script to set the file permissions according to the mode in which to run Oracle 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
```

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

```
webcache_setuser.sh command user_ID
```

Table 5–1 describes the commands.

Table 5–1 Commands of the webcache_setuser.sh Script

Command	Description
`setroot`	Sets the ownership of the `webcached` executable to root, and runs Oracle Web Cache as the user that performed the installation.
`setidentity`	Changes the ownership of the run time Oracle 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 cannot 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 Oracle 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 the Process Identity page (**Properties** > **Process Identity**) of Oracle Web Cache Manager. See the Section 5.9 for further information about when running the `webcache_setuser.sh` script is necessary.

6

Caching and Compressing Content

This chapter introduces techniques to cache and compress content using Oracle Web Cache. It discusses cache population, consistency, and rules. The chapter describes the properties for configuring caching rules and expiration policies.

This chapter includes the following topics:

6.1 About Cache Population

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

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

See Section 6.3 for a description of how Oracle Web Cache determines cache population through caching rules.

6.2 About Cache Consistency

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

- Section 6.2.1, "Expiration"
- Section 6.2.2, "HTTP Cache Validation"
- Section 6.2.3, "Invalidation"

6.2.1 Expiration

With **expiration**, Oracle 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 changes on an origin server or database. To prevent objects from remaining in the cache indefinitely, Oracle recommends creating expiration policies for all cached objects.

For instructions on creating expiration policies, see Section 6.7.

6.2.2 HTTP Cache Validation

Oracle 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 to determine if they represent the same or different entities. Specifically, Oracle Web Cache uses the `If-Modified-Since` and `If-None-Match` headers to determine the following validity:

- Browser header is valid as compared with the cached copy's header
- Cached copy's header is valid as compared with the origin server's header

---

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

For further information about validation, see:

- 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
- Section 6.2.3 for instructions on invalidating content
- Section 6.7 for instructions on configuring expiration policies

6.2.3 Invalidation

You use invalidation for content that does not have predictable expiration times. With **invalidation**, Oracle 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.

For further information about invalidation, see Chapter 7, "Invalidating Content."

6.3 About Caching Decisions

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 or the `Content-Type` response header with Fusion Middleware Control, or you set the caching attributes for a specific object within a `Surrogate-Control` response-header field. Those objects matching the rule are not cached until there is a client request for them.

Oracle Web Cache uses the following priority to determine object cacheability:

1. `Surrogate-Control` response header
2. Caching rule configured with Fusion Middleware Control
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 Oracle 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 Fusion Middleware Control. When both a `Surrogate-Control` response header and a caching rule for the same object are present, Oracle Web Cache merges the two. For example, if there is a caching rule for an non-cacheable object set in Fusion Middleware Control with compression enabled, and the response header contains `Surrogate-Control: max-age=30+60`, then Oracle Web Cache respects both settings. Oracle 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 Oracle Web Cache uses the settings from the `Surrogate-Control` response header.

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

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

For a description of how Oracle Web Cache determines cache population, see Section 6.1.

6.4 Introduction to Creating Caching Rules

When you decide to create a caching rule for an object, determine first whether the rule for the object is for a specific site or global to all sites. Oracle Web Cache gives site-specific caching rules a higher priority than the global rules. After you determine that choice, you configure general attributes for the rule:

- Information about the rule, such as its name, description, site, and whether or not to enable the rule
- Actions for Oracle Web Cache to take when an incoming request matches a rule
- Match criteria for Oracle Web Cache to locate the matching rule for incoming requests.

Select to base the match evaluation on the request URL expression, the response `MIME` type, or both criteria. If you do not select a match criteria, Oracle Web Cache matches the rule to all URLs and all `MIME` types.

Matching the evaluation on the `MIME` type makes sense when entering URL expressions becomes cumbersome. For example, the following shows a complicated URL expression for various image types:

```
\.(gif|jpe?g|png|bmp)$
```

Instead, you can select the **MIME Type** option, select **Starts with**, and enter the following string in the expression field:

```
image/
```

For most match evaluations, select the **Starts with** option, because the `Content-Type` response header typically has additional content parameter values.

You can also enter a list in the expression field, separated by commas. Continuing with the same example, if you want to match only GIF or JPEG responses, enter the following string in the expression field:

```
image/gif, image/jpeg
```

After you create caching rules, you order the priority of caching rules. Higher priority rules are matched first. Oracle Web Cache gives site-specific caching rules a higher priority than the global caching rules. When ordering caching rules for cacheable and non-cacheable objects, give the non-cacheable objects a higher priority than the cacheable objects.

In the rules shown in Table 6–1, 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`. If the order were reversed, all objects starting with `/cec/cstage?ecaction=ecpassthru` would be cached, including `/cec/cstage?ecaction=ecpassthru2`.

Table 6–1 Example of Priority for Different HTTP Methods

Priority	Match Criteria	HTTP Methods	POST Body Expression	Action

1	URL Expression:
	Regular Expression:
	`^/cec/cstage\?ecaction=ecpassthru2`
	Path Prefix:
	`/cec/cstage\?ecaction=ecpassthru2`
		GET and GET with query string	N/A	Don't Cache

2	URL Expression:
	Regular Expression:
	`^/cec/cstage\?ecaction=ecpassthru.*`
	Path Prefix:
	`/cec/cstage\?ecaction=ecpassthru`
		GET and GET with query string	N/A	Cache

3	URL Expression:
	Regular Expression:
	`^/cec/cstage\?ecaction=ecpassthru.*`
	Path Prefix:
	`/cec/cstage\?ecaction=ecpassthru`
		POST Body Expression	`action=search`	Cache

4	MIME Type: `/image`	GET and GET with query string	N/A	Cache

For more information about specifying general attributes for caching rules and specifying priority, see Section 6.8.1

6.5 Introduction to Configuring Advanced Settings

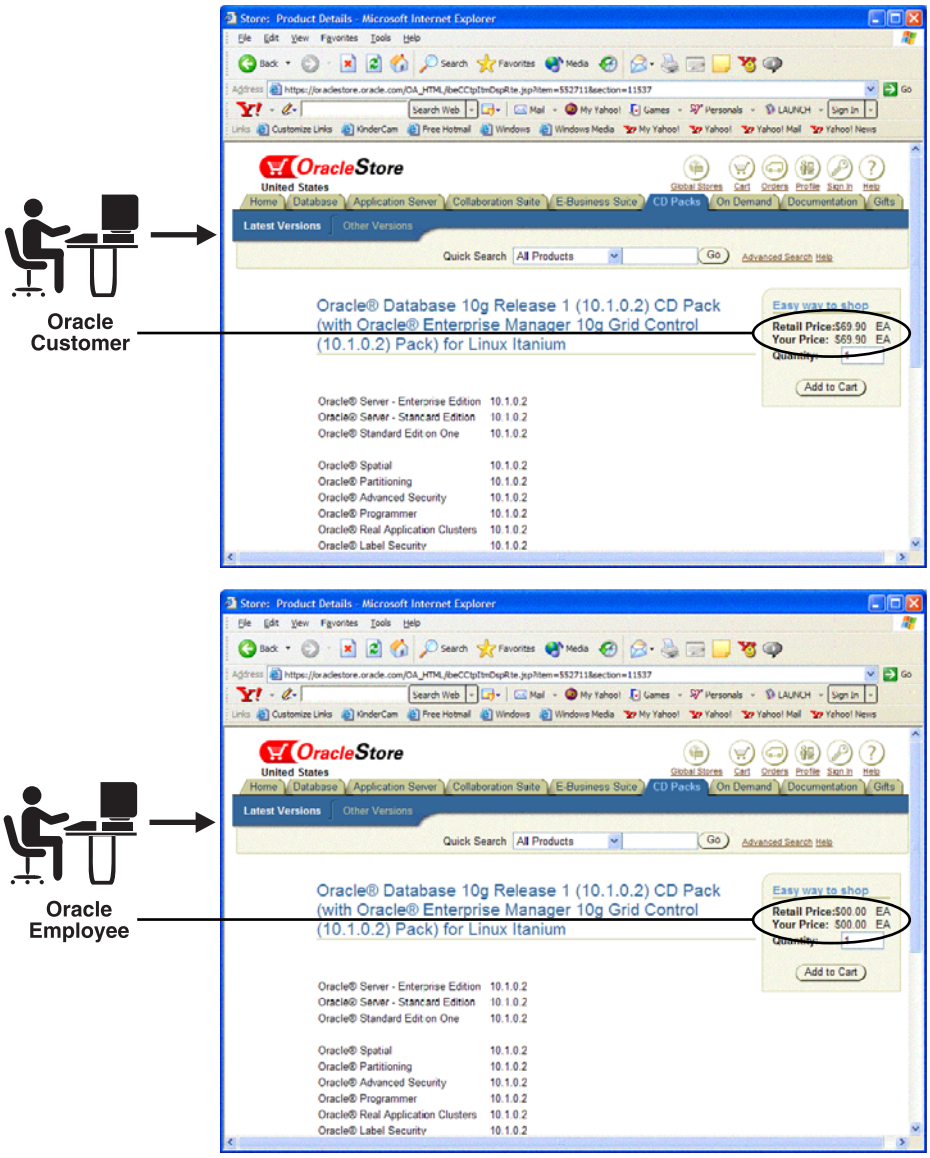
In addition to general attributes, you can configure advanced settings, as described in the following topics:

- Section 6.5.1, "Caching for Objects with Multiple Versions"
- Section 6.5.2, "Caching for Objects with Embedded URL and POST Body Parameters"
- Section 6.5.3, "Caching Error Responses"
- Section 6.5.4, "Caching for Objects with Sessions"
- Section 6.5.5, "Caching for Objects with Session-Encoded URLs"

6.5.1 Caching for Objects with Multiple Versions

Some pages have multiple versions, enabling categorization. Figure 6–1 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 6–1 Multiple-Version Object



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

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

For those 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, Oracle 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
```

Oracle Web Cache does not cache this initial response. 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`*

Oracle Web Cache still forwards the request to the origin server, which responds with or without the `Set-Cookie` header. Oracle Web Cache then 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. Oracle Web Cache consider the absence of the `Set-Cookie` header a match. If cookie and its value do not match, then the response is not cached. After versions of the object are cached, Oracle 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:** Oracle Web Cache does not cache the `Set-Cookie` response header field.

Table 6–2 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 Oracle Web Cache to recognize the `user_type` cookie, enabling Oracle Web Cache to cache three different objects. In addition, you can configure Oracle Web Cache to cache a fourth object for those requests that do not use a cookie.

Table 6–2 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. Oracle Web Cache uses the header to serve the appropriate version of the URL to clients.

Oracle Web Cache supports all valid HTTP request headers. Table 6–3 lists the HTTP request-header fields supported by Fusion Middleware Control. You can specify any of the standard or other HTTP request-header fields with the `Surrogate-Control` response-header field.

Table 6–3 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`

Table 6–3 (Cont.) HTTP Request-Header Field

Header Field	Description
`Accept-Encoding`	Restricts the content-encodings that are acceptable in the response **Example:** `Accept-Encoding: gzip`
`Accept-Language`	Specifies the set of languages that are preferred as a response **Example:** `Accept-Language: en`
`User-Agent`	Contains information about the client that initiated the request **Example:** `User-Agent: Mozilla/4.61 [en] (WinNT; U)`

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

This issue is especially problematic with the `User-Agent` request header, whereby the browser type, version, and operating system can result in too many duplicate 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, Oracle Web Cache serves two pages for the two requests.

You can override this behavior for the `User-Agent` request header by configuring Oracle Web Cache to cache and serve the same page for the same browser type, as described in Section 6.8.4.

For configuration details, see Section 6.8.2.

6.5.2 Caching for Objects with Embedded URL and POST Body Parameters

By default, Oracle 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, Oracle Web Cache caches responses for each of the distinct URLs. This can result in low cache hit rates and redundantly cached objects.

By configuring Oracle Web Cache to ignore the value of embedded URL or POST body parameters, you enable Oracle Web Cache to serve one cached object to multiple sessions requesting the same page. Oracle 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 Oracle 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 to be applied to all caching rules or site-specific parameters to be applied to caching rules for a specific site. See Section 6.8.3.

6.5.3 Caching Error Responses

If there is a problem on the **origin server** that does not result in a 200 OK HTTP response status for a request that matches this rule, then Oracle Web Cache does not attempt to send the request to the origin server again. Instead, it serves the cached HTTP error, saving origin server resources for known bad responses.

By default, Oracle Web Cache does not cache any non-200 OK HTTP responses. If you want these errors to be cached, then you must configure caching rules to specifically cache error responses. Oracle Web Cache caches the error pages according to the expiration policy of the rule. After the problem is resolved, invalidate the HTTP error responses.

For configuration details, see Section 6.8.5.

6.5.4 Caching for Objects with Sessions

You can specify how Oracle 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. Oracle Web Cache uses a default string for those requests without the cookie or parameter.

For configuration details, see Section 6.8.6.

6.5.5 Caching for Objects with Session-Encoded URLs

The section Section 6.5.2 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 Oracle 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 Oracle Web Cache to cache one object for multiple sessions, even if the session parameter values in session-encoded URLs vary.

Note: Oracle Web Cache does not cache the `Set-Cookie` response header field.

Continuing with the example from Section 6.5.2, 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 6–2 has several `<A HREF=...>` links that include the `session_ID` parameter. The **Oracle Database Standard Edition** link under the **Oracle Database** heading for Jane Doe uses the following HTML code:

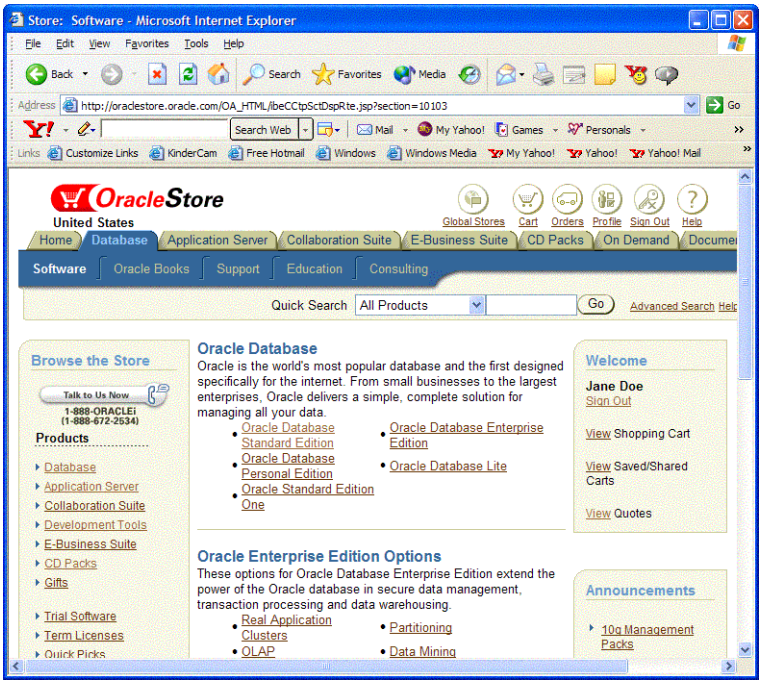
```
<A HREF="http://oraclestore.oracle.com/OA_
HTML/ibeCCtpSctDspRte.jsp?section=10166&session_ID=334326">Oracle Database
Standard Edition</A>
```

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">Oracle Database
Standard Edition</A>
```

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

Figure 6–2 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, you can configure Oracle Web Cache to substitute the session information in the session-encoded URLs with a configurable default string.

For configuration details, see Section 6.8.7.

6.6 Basic Tasks for Configuring and Monitoring Caching Rules

The following provides a summary of the steps required to cache and monitor objects:

1. Configure expiration policies. See Section 6.7.
2. Create sites for which Oracle Web Cache manages requests. See Section 2.11.3 and Section 2.11.4.
3. Configure general settings for a caching rule. See Section 6.8.1.
4. Configure advanced settings for a caching rule:
   - Configure cookie and HTTP request-header fields for rules supporting multiple-version objects. Section 6.8.2.
   - Configure Oracle Web Cache to ignore the value of embedded URL or POST body parameters. See Section 6.8.3.
   - Configure error responses for rules. See Section 6.8.5.
   - Configure session settings for rules. See Section 6.8.6.
   - Configure support for session-encoded URLs. See Section 6.8.7.
5. Restart Oracle Web Cache to apply caching rule. See Section 2.13.
6. Monitor statistics for caching rules. See Section 6.9.
7. View popular requests. See Section 8.2.

6.7 Configuring Expiration Policies

Prior to creating a caching rule, you create expiration policies. Later, when you create caching rules, you specify an expiration policy to apply with the caching rule.

You can create expiration 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, Oracle Web Cache removes it either immediately or as permitted by origin server capacity up to a maximum time limit.

To create an expiration policy:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Expiration**.

   The Expiration Policies page displays.
3. Click **Create**.

   The Create Expiration Policy dialog displays.
4. In the **Objects Expire** section, specify when to expire objects by selecting an option:
   - **As per HTTP Expires Header:** Select this option to respect the HTTP `Cache-Control` or `Expires` response-header fields. This is the default.

- **After Cache Entry:** Select this option to base expiration on when the objects entered the cache. Enter the time to expire the objects in the **Time Limit** field.
- **After Creation:** Select this option to base expiration on when the objects were created, as indicated by the origin server. Enter the time to expire the objects in the **Time Limit** field.

5. In the **Action On Expired Objects** section, specify how you want Oracle Web Cache to process objects after they have expired:
   - **Remove Immediately:** Oracle 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 Origin Server Permits:** Oracle Web Cache marks objects as stale and then refreshes them based on origin server capacity. Oracle Web Cache may serve the stale content when the origin server is heavily loaded. Enter the maximum time in which the objects can reside in the cache and be served stale in the **Time Limit** field.
6. Click **OK** to apply changes.
7. Restart Oracle Web Cache. See Section 2.13.

6.8 Configuring and Monitoring Caching Rules

This section describes how to configure caching rules for Oracle Web Cache. It includes the following topics:

6.8.1 Configuring General Rule Settings

Before you create a caching rule, determine first whether the rule for the object is for a specific site or global to all sites. Oracle Web Cache gives site-specific caching rules a higher priority than the global rules.

For more information about caching decisions, see Section 6.3.

To create a caching rule:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Caching Rules**.

   The Caching Rules page displays.
3. Create a new rule.

a. From **Site Specific Caching Rules** or **Global Caching Rules** section, click **Create**.

   The Create Caching Rule page displays with the **General** tab in view.

b. Complete the elements using the descriptions in Table 6–4.

   When completing the elements, in the **Match Criteria** section, select to base match evaluation on the request URL expression, the response `MIME` type, or both criteria. If you do not select a match criteria, Oracle Web Cache matches the rule to all URLs and all `MIME` types.

   If you find entering URL expressions is cumbersome for your rules, select the **MIME Type** option in the **Match Criteria** section. See Section 6.4 for further information about using the **MIME Type** option in place of complicated URL expressions.

c. Click **OK** to apply changes and return to the Caching Rules page. It is not necessary to click **Apply** in the Create Caching Rule page to apply this change.

   Notice the caching rule is added to the **Site Specific Caching Rules** or **Global Caching Rules** table.

4. Repeat Step 3 for each additional rule.

5. Use the **Move Up** and **Move Down** icons to change the order in which the rules are matched against requests.

   The order of the rules is important. Oracle Web Cache matches higher priority rules first.

6. Click **Apply** to apply the move change.

7. In the Caching Rules page, click **Enable** to enable rules.

   If you do not click **Enable**, Oracle Web Cache ignores any the settings for the rule.

8. Restart Oracle Web Cache. See Section 2.13.

Table 6–4 Caching Rules - General Page

Element	Description
Name	Enter a string that uniquely identifies the caching rule.
Description	Enter a descriptive comment about the caching rule.
Enabled	Select to enable the caching rule; deselect to disable the caching rule temporarily without losing the rule definition.
Site	Displays the site for which to apply this rule. If you do not see the site required, create one, following the procedure in Section 2.11.3.
Cache	Select this option to instruct Oracle Web Cache to cache content; deslect this option to instruct Oracle Web Cache to forward requests to the origin server and to not cache the content.
Expiration	From the list, select an expiration policy to apply to the objects. If you do not see an expiration policy suitable for the objects, click the **Expiration Policies** link.

Table 6–4 (Cont.) Caching Rules - General Page

Element	Description
Compress	Select this option to instruct Oracle Web Cache to serve compressed cacheable and non-cacheable objects to browsers. To enable compression for this rule, you must also enable compression for the site. To set the compression property for a site, see Section 2.11.3. Oracle Web Cache automatically disables compression for some common file types which are known to be already compressed. Oracle recommends not compressing content for these file type, including GIF, JPEG, and PNG images, or files that are already compressed with utilities like WinZip or GZIP. Compressing these files incurs additional overhead without gaining any compression benefit. See Section 1.2.5 to better understand when Oracle Web Cache automatically disables compression.
Match URL By	Select to base the match evaluation on the URL expression: 1. Select the expression type: **- File Extension:** Select to apply the caching rule to objects ending in a particular file extension, such as `.gif`. In the accompanying field, enter the file extension. Because Oracle Web Cache internally starts the file extension with a period (`.`), it is not necessary to enter it. **- Path Prefix:** Select to apply the caching rule to objects matching a path prefix. In the accompanying field, enter the path prefix of the objects. Start the path with `/`; do not start the path with `http://`*`host_name`*`:`*`port`*`/`. The prefix is interpreted literally, including reserved regular expression characters. These characters include periods (`.`), question marks (`?`), asterisks (`*`), brackets (`[]`), curly braces (`{}`), carets (`^`), dollar signs (`$`), and backslashes (`\`). **- Regular Expression:** Select to apply the caching rule to objects matching regular expression syntax. In the accompanying field, enter the regular expression of the objects. Remember to use "^" to denote the start of the URL and "$" to denote the end of the URL. 2. Click the **Case Insensitive Match** check box to match requests regardless of the case. If you do not select this check box, the rule bases the match on the case.

Table 6–4 (Cont.) Caching Rules - General Page

Element	Description
MIME Type	Select to base the match evaluation on the `Content-Type` response header: 1. Select an operation value to determine where Oracle Web Cache searches for the expression value in the response's `MIME` type: **-Equals:** Select to instruct Oracle Web Cache to match the `MIME` type if it equals the expression. **-Starts With:** Select to instruct Oracle Web Cache to match the `MIME` type if it starts with the expression. **-Contains:** Select to instruct Oracle Web Cache to match the `MIME` type if it contains the expression. For most match evaluations, select the **Starts with** option, because the `Content-Type` response header typically has additional content parameter values. If you select **Equals**, then the match may not work as you expect. 2. In the accompanying field, enter the string value that you want Oracle Web Cache to compare against the response's `MIME` type. You can enter a list in the expression field, separated by commas.
HTTP Methods	Select one or more of the following HTTP request methods: ■ **GET:** 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 methods are typically cached. ■ **GET with query string:** An HTTP request method made up of a URL and a query string containing parameters and values. ■ **POST Body Expression:** An HTTP request method used for requests that modify the contents of the data store on the application Web server, such as posting a message to a mailing list, submitting forms for registration purposes, or adding entries to the database. **Note:** 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 Body Expression**. Select these options only if the forms are used in search forms.
Required Request Parameters	Click **Add** to enter an embedded URL parameter or POST body parameter and its value in the corresponding **Parameter Name** and optional **Value** fields. **Notes:** ■ If you selected **Regular Expression** from the **Match URL By** list, you either use the **Required Request Parameters** section or manually enter the embedded URL parameters alphabetically in the accompanying **Match URL By** field. When you use the **Required Request Parameters** section, the embedded URL parameters are automatically sorted. See Section 6.8.1.1 for information about how to specify parameters in the accompanying **Match URL By** field. ■ If you selected **POST** in the **HTTP Methods** section and do not specify POST body parameters in this section, specify the HTTP POST body in the accompanying **POST Body Expression** field. To apply this rule to any POST request body, enter `.*"` in the field.

6.8.1.1 Regular Expression Parameters

The request URL that client browsers send to Oracle Web Cache and the internal URL expression that Oracle Web Cache uses for that request are different. When Oracle 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 **Required Request Parameters** section or manually enter the embedded URL parameters alphabetically in regular expression syntax in the **Match URL By** field. When you use the **Required Request Parameters** section, the Oracle Web Cache automatically sorts the embedded URL parameters.

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 **Match URL By** expression field, `^/servlet/page\?_pageid=53&_dad=moc&_schema=MOC$`, then the caching rule does not match the internal representation of the URL used by Oracle 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$
```

6.8.2 Configuring Settings for Rules with Multiple Versions of the Same Object

For more information about caching multiple-version objects, see Section 6.5.1.

To specify a caching rule for multiple-version objects:

1. For sites with category cookies that affect caching, specify which category cookies whose values Oracle Web Cache uses to cache and identify multiple-version objects:
   a. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
   b. From the **Web Cache** menu, select **Administration** and then **Multi-Version Cookies**.

      The Multi-Version Cookie page displays.
   c. Click **Create**.
   d. In the **Cookie Name** field, enter the name of the cookie.
   e. Click the **Cache If Absent** check box to cache versions of the object that do not contain this cookie. This option enables Oracle Web Cache to serve objects from the cache for client requests that do not contain this cookie. Keep this option unchecked to not cache versions of objects that do not contain this cookie.
   f. Click **Apply**.
2. Create a caching rule. See Section 6.8.1.
3. From **Site Specific Caching Rules** or **Global Caching Rules** section of the Caching Rules page, select the rule you created and click **Edit**.

   The Create Caching Rule page displays.
4. Click the **Multi Versioning** tab.

5. For multiple-version objects that rely on a cookie or HTTP request header, specify the cookies or HTTP request headers to enable Oracle Web Cache to cache multiple versions of an object and serve the appropriate version to requests.

   For multiple-version objects with cookies, select the cookie you created in Step 1.

   For multiple-version objects with HTTP headers, from the **By HTTP Request Headers** section, select one or more of the headers from the **Available Headers** and click **Move** or **Move All** to move them to the **Selected Headers** list.

6. Click **OK** to apply changes.

7. Restart Oracle Web Cache. See Section 2.13.

6.8.3 Excluding the Value of Embedded URL or POST Body Parameters

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

For more information about configuring Oracle Web Cache to ignore the value of parameters can enable Oracle Web Cache to serve one cached object to multiple sessions, see Section 6.5.2.

You have two configuration options for specifying parameters to ignore. You can:

- Establish global parameters to be automatically applied to all global caching rules and site-specific caching rules.
- Specify site-specific parameters to be automatically applied to caching rules for that site.

To establish global parameters:

1. From Oracle Web Cache Manager, in the navigator frame, select **Origin Servers, Sites, and Load Balancing** > **Site Definitions**. See Section 2.7.2.

   The Site Definitions page displays.

2. Click **Edit Global URL Parameters to Ignore** to specify global parameters for all sites.

   The Global URL Parameters to Ignore dialog displays.

3. In the **Parameters to Ignore** field, specify the global parameters. Separate multiple parameters by commas or blank spaces.

4. Click **Submit**.

5. Click **Apply Changes**.

6. Restart Oracle Web Cache. See Section 2.13.

7. Create a site-specific caching rule. See Section 6.8.1

To establish site-specific parameters:

1. From Oracle Web Cache Manager, in the navigator frame, select **Origin Servers, Sites, and Load Balancing** > **Site Definitions**. See Section 2.7.2.

   The Site Definitions page displays.

2. Select a site, and then click **Show/Edit Selected**.

   The Show/Edit Site Definition dialog displays.

3. In the **URL Parameters to Ignore** field, specify the site-specific parameters. Separate multiple parameters by commas or blank spaces.
4. Click **Submit**.
5. Click **Apply Changes**.
6. Restart Oracle Web Cache. See Section 2.13.
7. Create a site-specific caching rule. See Section 6.8.1

6.8.4 Recognizing Similar Browser Types for Multiple-Version Objects Containing HTTP Request Headers

By default, Oracle 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 Fusion Middleware Control and the value of the `User-Agent` request header of the same URL differ, then Oracle Web Cache caches both pages separately. For example, if one request sends an HTTP request header of `User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows)` 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, Oracle Web Cache caches two separate pages.

You can override this default behavior by configuring Oracle Web Cache with a `User-Agent` pattern string for a particular client. For the affected multiple-version objects, Oracle 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
```

To configure Oracle 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 Section 6.8.2, ensuring you select the `User-Agent` header.
2. Use a text editor to open `webcache.xml`, located in:

   ```
   (UNIX) ORACLE_INSTANCE/<instance_name>/config/WebCache/<webcache_name>
   (Windows) ORACLE_INSTANCE\<instance_name>\config\WebCache\<webcache_name>
   ```

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 Oracle Web Cache to match. The order of these rules work in the same fashion as priority works for caching rules.

   Table 6–5 describes how to enter values for the subelements.

Table 6–5 GLOBALCACHINGRULES Subelements

Subelement	Description
MATCHSTRING	Enter the pattern that 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 to be added to the x-Oracle-Mapped-User-Agent request header by Oracle 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, Oracle 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 Oracle Web Cache using opmnctl. See Section 2.13.1.

6.8.5 Configuring Error Responses for Rules

To understand how you can cache HTTP error responses to save origin server resources, see Section 6.5.3.

To create a caching rule for an error response:

1. Create a caching rule. See Section 6.8.1.
2. From the **Site Specific Caching Rules** or the **Global Caching Rules** section of the Caching Rules page, select the rule you created and click **Edit**.

   The Create Caching Rule page displays.
3. Click the **Error Responses** tab.

4. Select the HTTP error codes you want Oracle Web Cache to cache and serve for this rule.

   Ensure the origin server generates the HTTP error itself.

5. Click **OK** to apply changes.

6. Restart Oracle Web Cache. See Section 2.13.

6.8.6 Configuring Session Caching Rules

To understand how Oracle Web Cache serves requests with the existence or nonexistence of session cookies, embedded URL parameters, or POST body parameters, see Section 6.5.4.

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

1. From the Web Cache menu, select **Administration** and then select **Session Configuration**.

   The Session Definitions page displays.

2. Create a session definition in the **Session Definitions** table. See Section 2.12.

3. Specify session policy settings:

   a. In the **Session Policy Configuration** section, click **Create**.

      A new row in the table appears.

   b. From the **Session Name** list, select the session you created in Step 2.

   c. In the **Cache** column, select the **Without Session** check box for Oracle Web Cache to cache versions of objects that do not use the cookie or parameter; select **No** for Oracle Web Cache not to serve objects from the cache for requests without the session information.

   d. In the **Cache** column, select the **With Session** check box for Oracle Web Cache to cache versions of objects that use the cookie or parameter.

   e. In the **Substitute Default Value** column, select the check box to instruct Oracle Web Cache to cache one version of the object. For those requests without a cookie or parameter, a default value is used. Do not select the check box to instruct Oracle Web Cache to cache two different versions of the object. Oracle 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.

4. Create a caching rule. See Section 6.8.1.

5. Associate session policies with a caching rule:

   a. From the **Site-Specific Caching Rules** or the **Global Caching Rules** section of the Create Caching Rule page, select the rule you created and click **Edit**.

      The Edit Caching Rules page displays.

   b. Click the **Sessions** tab.

   c. From the **Session Definition** list, select the sessions you created in Step 2 and defined a policy for in Step 3.

   d. Click **OK** to apply changes.

6. Restart Oracle Web Cache. See Section 2.13.

6.8.7 Configuring Support for Session-Encoded URLs

You can configure Oracle 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. To understand how Oracle Web Cache can cache one object for multiple sessions, even if the session parameter values in session-encoded URLs vary, see Section 6.5.5.

To substitute session values in session-encoded URLs:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the Web Cache menu, select **Administration** and then select **Session Configuration**.

   The Session Definitions page displays.
3. Create a session definition in the **Session Definitions** table. See Section 2.12.

   When entering data for the **Default Value** field, enter a default string for the value of the embedded URL parameter.

   Oracle Web Cache uses the value you enter in the **Default Value** field for those requests without the value for an embedded URL parameter. For these requests, Oracle 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, Oracle 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>
   ```
4. Create a caching rule. See Section 6.8.1.
5. From the **Site Specific Caching Rules** or the **Global Caching Rules** section of the Caching Rules page, select the rule you created and click **Edit**.

   The Create Caching Rule page displays.
6. Click the **Sessions** tab.
7. Click **Process for Session-Encoded URLs**.
8. Click **OK** to apply changes.
9. Restart Oracle Web Cache. See Section 2.13.

6.8.8 Configuring Rules for Popular Pages with Session Establishment

Some Web sites require users to have sessions while surfing most pages. To preserve the session requirement, create a session caching rule for those pages. This way, Oracle Web Cache always forwards a request without a session to 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 session caching rule for the real entry page and the blank page, as described in Section 6.8.6, and click both the **Cache With** and **Cache Without** options.

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 Section 6.8.1 and Section 6.8.6.

Note: Using the Javascript solution, it is not necessary to create a session caching rule for the pages requiring sessions.

6.9 Monitoring Summary Settings for Caching Rules

Fusion Middleware Control provides statistics for assessing the effectiveness of configured caching rules.

To view caching-rule statistics:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Caching Rules**.

   The Caching Rules page displays.
3. Scroll to the far right to view the statistics for a caching rule:
   - **Multi-Versionings:** Shows whether this caching rule contains settings for requests with multiple versions.
   - **Sessions:** Shows whether this caching rule contains settings for requests with a cookie, embedded URL parameter, or POST body parameter with user session information.
   - **Error Responses:** Shows whether this caching rule contains settings for requests with HTTP error responses.
   - **Matched:** Displays the number of requests that matched the caching rule.

See Section 8.2 to view the objects cached by Oracle Web Cache

6.10 Using the Surrogate-Control Response Header as an Alternative to Caching Rules

In addition to, or as an alternative to, creating caching rules with Fusion Middleware Control, 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 Fusion Middleware Control interface, as well as allow other third-party caches to use Oracle Web Cache caching attributes. All except the following attributes described in Section 6.8 are supported:

- Session caching rules
- HTTP error response caching rules

To enable this feature, set the HTTP response with the `Surrogate-Control` response-header field as described in the following section.

For a description of how Oracle Web Cache uses caching attributes from the `Surrogate-Control` response header and Oracle Web Cache Manager to determine cache population, see Section 6.1.

6.10.1 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 Fusion Middleware Control or Oracle 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...)][cookie(cookie_name cookie_name...)]
[compress=yes|no]
```

Table 6–6 describes the supported control directives.

Table 6–6 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 Oracle 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 Oracle 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. For further information about the ESI tags supported for each processing version, see Chapter 11, "Caching Dynamic Content with ESI Language Tags."
`no-store`	Specify for Oracle Web Cache to not cache the object.
`vary`	Specify the HTTP request headers or cookies to instruct Oracle Web Cache to cache and identify multiple-version objects. Use the following format: `vary=[headers(header[/f] *);] [cookies(cookie_name[/f] *)]` Specify `/f` to instruct Oracle Web Cache to only cache versions of the object based on the existence of the HTTP request headers or cookies. Exclude `/f` to instruct Oracle 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.
`compress`	Specify `yes` for Oracle Web Cache to serve compressed cacheable and non-cacheable objects to all browser types; specify `no` for Oracle 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 must limit the browser types, specify a compression caching rule, as described in Section 6.8.1. To understand what Oracle Web Cache automatically compressed and does not compress, see Section 1.2.5.
`max-age`	Specify for Oracle Web Cache to 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"`

Refer to `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 based on the HTTP `Accept` request header value, 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
MyCustomHeader/f);cookies(news/f sports/f)
```

7 Invalidating Content

This chapter explains how to send invalidation requests to Oracle Web Cache.

This chapter includes the following topics:

7.1 Overview of Invalidation

As described in Section 6.7, you create expiration policies and associate them with caching rules to refresh content from the origin server. Even with expiration policies, it is often difficult to predict when exactly content becomes stale. As an alternative, Oracle Web Cache provides mechanisms for explicitly invalidating content when an administrator or application knows that such content has become stale.

With invalidation, Oracle 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.

Oracle Web Cache supports the following forms of invalidation:

- Invalidation through a special invalidation port. This type of invalidation is known as an out-of-band invalidation, because such invalidation requests do not go through the Oracle Web Cache listening port. See Section 7.2.
- ESI inline invalidation where the invalidation directive comes as an ESI tag through the response body to a normal request. See Section 7.3.

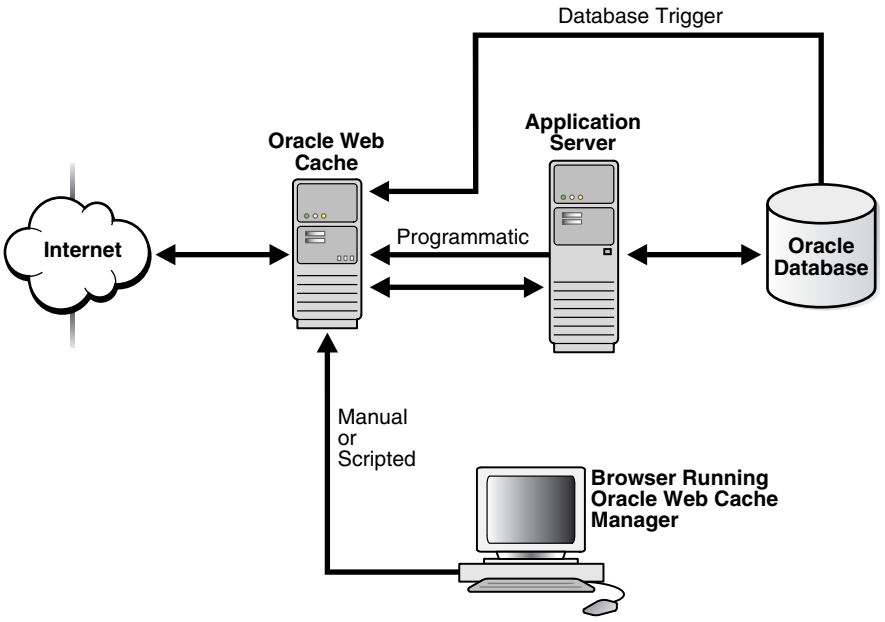
- Response-header invalidation where the invalidation directive comes as a special (and proprietary) response header in the response to a normal request. See Section 7.4.

7.2 About Out-of-Band Invalidations

To invalidate objects in the cache, you can send an HTTP POST request from the `invalidator` account through the invalidation listening port. The `invalidator` account is authorized to send invalidation requests. As shown in Figure 7–1, you send invalidation requests using these methods:

- Manually, using Fusion Middleware Control or `telnet`
- Automatically, using database triggers, scripts, or application logic

Figure 7–1 Invalidation



The following sections describe the specific methods you can use:

- Section 7.7.1, "Using Telnet to Send Invalidation Requests"
- Section 7.7.2, "Using Oracle Web Cache Manager to Send Invalidation Requests"
- Section 7.7.3, "Using Application Program Interfaces (APIs) for Automated Invalidation Requests"
- Section 7.7.4, "Using Database Triggers for Automated Invalidation Requests"
- Section 7.7.5, "Using Scripts for Automated Invalidations"

7.3 About ESI Inline Invalidations

Inline invalidation is implemented as part of **Edge Side Includes (ESI)** and provides a useful way for origin servers to "piggyback" invalidation messages on HTTP responses sent to Oracle Web Cache. Specifically, origin servers embed an XML invalidation document within the HTML of the response body using ESI tags.

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.

For more information about using ESI invalidation, see Section 11.3.

7.4 About Response Header Invalidations

Response header invalidation is Oracle Web Cache functionality that enables an origin server to return a transactional response whose response body contains something other than HTML. This is a circumstance in which ESI inline invalidation does not work; Oracle Web Cache can only use ESI invalidation tags in conjunction with a response body that contains HTML. With response header invalidation, origin servers can send invalidation directives in a proprietary invalidation response header.

In addition to its greater flexibility in terms of response body content returned, response header invalidation requires less coding effort on the part of the Web applications since building an invalidation header is a fairly lightweight task.

Response header invalidation functions similarly to inline invalidation; origin servers "piggyback" invalidation directives on responses sent to Oracle Web Cache. However, the response header invalidation enables invalidation when the response body contains something other than HTML.

The origin server adds a special invalidation header to its response. Oracle Web Cache extracts the invalidation header, invalidates the corresponding content and forwards the response to its client but without the invalidation header in the response.

Origin Servers can piggyback an invalidation response header on any random response. For example, an origin server may delay the sending of an invalidation directive and then send it later in a response to a request that has nothing to do with the request that caused the invalidation in the first place.

Oracle Web Cache strips out the invalidation response header when returning the response to a Web client. Oracle Web Cache even strips out the invalidation response header when returning the response to another member of an cache cluster, since cluster propagation forwards the invalidation to other peers in the cluster.

For more information about enabling response header invalidation, see Section 7.8.

7.5 Format of Invalidation Requests for Out-of-Band and ESI Inline Mechanisms

The out-of-band mechanisms send out-of-band HTTP POST invalidation requests in **Extensible Markup Language (XML)** syntax. The ESI inline mechanism also uses the same syntax within the `<esi:invalidate>` tag. The contents of the XML request body instructs the cache which URLs to mark as invalid.

The following sections describe invalidation request syntax:

- Section 7.5.1, "Invalidation Request Syntax"
- Section 7.5.2, "Invalidation Response Syntax"
- Section 7.5.3, "Invalidation Preview Request Syntax"
- Section 7.5.4, "Invalidation Preview Response Syntax"
- Section 7.5.5, "Invalidation Examples"

7.5.1 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 7–1.

Table 7–1 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 is not 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 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 multiple OBJECT elements in the request.
BASICSELECTOR element	URI attribute: Required attribute of the BASICSELECTOR element. Specify the URL of the objects to be invalidated. Use these 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 7–1 (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 are matched. Regular expression characters are permitted. To interpret these characters literally, escape them with a backslash (\). See http://www.cs.utah.edu/dept/old/texinfo/regex/regex_toc.html for regular expression syntax. **Notes:** ■ The request URL that client browsers send to Oracle Web Cache and the URL that Oracle Web Cache uses internally for that request are different. When Oracle 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, Oracle 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 either GET or POST for the **HTTP request method** of the objects to be invalidated. GET is the default value. BODYEXP attribute: Optional attribute of the ADVANCEDSELECTOR element. If the METHOD is set POST, specify the HTTP POST body of the objects to be invalidated. **Note:** When the invalidation request is sent, Oracle 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. For more information, see: ■ Section 6.8.2 to create caching rules for multiple-version objects ■ Section 6.8.6 to specify session caching policies

Table 7–1 (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** to invalidate multiple-version objects based on the request header. ■ VALUE attribute: Optional attribute for the HEADER element. Specify the value of the header. See Section 6.8.2 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 with a backslash (\) for Oracle Web Cache to interpret literally. For more information, see: ■ Section 7.5.5 to optimize advanced invalidations ■ Section 7.9 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, writes to the event log: `[15/Oct/2008:19:26:46 +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: Oracle 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>
```

7.5.2 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 7–2. `BASICSELECTOR` and `ADVANCEDSELECTOR` are described in Table 7–1.

Table 7–2 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.
`SYSTEMINFO` element	Required element in the `SYSTEM` element. The possible `NAME/VALUE` pairs is as follows: `NAME="WCS_CACHE_NAME" VALUE="`*`string`*`"` This pair specifies the name of the cache.
`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: - `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

7.5.3 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 7–3. `BASICSELECTOR` and `ADVANCEDSELECTOR` are described in Table 7–1.

Table 7–3 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. Oracle 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, Oracle Web Cache lists only the URLs for those objects that meet the criteria. If more objects than the number specified meet the invalidation criteria, Oracle 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.

7.5.4 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 7–4. `BASICSELECTOR` and `ADVANCEDSELECTOR` are described in Table 7–1.

Table 7–4 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: ■ `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

7.5.5 Invalidation Examples

This section contains the following invalidation request examples:

- Section 7.5.5.1, "Example: Invalidating One Object"
- Section 7.5.5.2, "Example: Invalidating Multiple Objects"
- Section 7.5.5.3, "Example: Invalidating a Subtree of Objects"
- Section 7.5.5.4, "Example: Invalidating All Objects for a Web Site"
- Section 7.5.5.5, "Example: Invalidating Objects Using Prefix Matching"
- Section 7.5.5.6, "Example: Invalidating Objects Using Substring and Query String Matching"
- Section 7.5.5.7, "Example: Invalidating Objects Using Search Key Matching"

- Section 7.5.5.8, "Example: Propagating Invalidation Requests Throughout a Cache Cluster"
- Section 7.5.5.9, "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.

7.5.5.1 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.

7.5.5.2 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:

```
[15/Oct/2008:19:26:46 +0000] [notification 11748] [invalidation] [ecid:
 21085932167,0] Invalidation with INFO 'summary.jsp' has returned with status
 'SUCCESS'; number of objects invalidated: '2'.
.
.
.
[15/Oct/2008:19:26:46 +0000] [notification 11748] [invalidation] [ecid:
 21085932167,0] Invalidation with INFO 'summary.gif' has returned with status
 'SUCCESS'; number of objects invalidated: '14'.
```

7.5.5.3 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>
```

7.5.5.4 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>
```

7.5.5.5 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 Oracle 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 with a URI containing `/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.

7.5.5.6 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 writes to the event log:

```
[15/Oct/2008:19:26:46 +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: '52'.
```

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 writes to the event log:

```
[15/Oct/2008:19:26:46 +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 Section 7.7.2.2 to optimize invalidations using `QUERYSTRING_PARAMETER`.

7.5.5.7 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>
```

7.5.5.8 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 in Fusion Middleware Control and Oracle Web Cache Manager, as described in Section 3.6.5 and Section 3.7.4, respectively.

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 Fusion Middleware Control or Oracle Web Cache Manager. If `NAME` is set to `WCS_PROPAGATE` and `VALUE` is set to `FALSE`, it reads the setting specified in Fusion Middleware Control or Oracle 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="/web_cache_host_name: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>
```

7.5.5.9 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>
```

7.6 About Search Keys in Invalidations

You 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 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, Oracle 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.
- Space between search keys is optional.

- The search keys must remain 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=( )
```

For more information about enabling search-key invalidation, see Section 7.9.

7.7 Initiating Out-of-Band Invalidations

The following topics describe how to initiate out-of-band invalidations:

- Section 7.7.1, "Using Telnet to Send Invalidation Requests"
- Section 7.7.2, "Using Oracle Web Cache Manager to Send Invalidation Requests"
- Section 7.7.3, "Using Application Program Interfaces (APIs) for Automated Invalidation Requests"
- Section 7.7.4, "Using Database Triggers for Automated Invalidation Requests"
- Section 7.7.5, "Using Scripts for Automated Invalidations"

7.7.1 Using Telnet to Send Invalidation Requests

When you send an invalidation request with an HTTP POST request, you specify the host name of Oracle 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 Oracle 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.

For more information, see:

- http://www.rfc-editor.org/ for information about password Base64 encoding
- readme.examples.htmlfor further information about using the EncodeBase64.java script to generate the Base64 string for invalidator:*invalidator_password*. This file is located in the following directories:

  (UNIX) *ORACLE_HOME*/webcache/docs
  (Windows) *ORACLE_HOME*\webcache/docs

- Section 5.2 for further information about changing the invalidation password

3. Enter one carriage return.
4. Send the invalidation request with XML syntax, as specified in Section 7.5.

7.7.2 Using Oracle Web Cache Manager to Send Invalidation Requests

Oracle Web Cache Manager provides an easy-to-use interface for invalidating cached objects. The advantage of using this 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 how quickly those objects should be invalidated.

Oracle 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:

- Section 7.7.2.1, "Submitting Basic Invalidation Requests"
- Section 7.7.2.2, "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 Oracle 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 processes. Until you restart the cache server process for either configuration change, invalidation requests return the error.

See Section 2.13 for restart instructions.

7.7.2.1 Submitting Basic Invalidation Requests

To send a basic invalidation request using Oracle Web Cache Manager:

1. From Oracle Web Cache Manager, in the navigator frame, select **Operations** > **Basic Content Invalidation.** See Section 2.7.2.

   The Basic Content Invalidation page appears in the right pane.

2. From the **For Cache** list, select a cache. (The list displays multiple caches 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 propagates 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 Oracle 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. Oracle Web Cache begins the count of objects with the number 0.
      – **To:** Enter the number of objects to be listed.

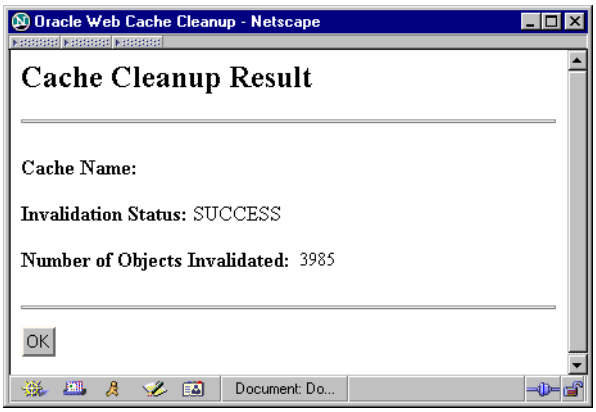
        If fewer objects than the number specified meet the invalidation criteria, Oracle Web Cache lists the URLs for only those objects that meet the criteria.

        If more objects than the number specified meet the invalidation criteria, Oracle 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 additional objects, send another preview request with a different **From** number that specifies the start of the next set of objects.
   c. Click **Submit.**

      Oracle Web Cache displays the Invalidation Preview Results message box, which lists the objects that meet the invalidation criteria. Oracle 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 listed objects are for those 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 an option to specify how to process invalid objects:
   - **Remove immediately:** Oracle 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:** Oracle 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. When you select this option, Oracle Web Cache applies heuristics to determine when is best time to no longer serve the document.
6. Click **Submit.**

Oracle Web Cache processes the invalidation request, and returns the Cache Cleanup Result dialog box which shows the invalidation status. The following figure shows the dialog box:



In a cache cluster environment, if **Invalidation requests sent to any cluster member will be propagated to all cluster members** is enabled, Oracle 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, Oracle 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.

For more information, see Section 3.6.5 for information about enabling invalidation propagation.

7.7.2.2 Submitting Advanced Invalidation Requests

To send an advanced invalidation request using Oracle 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. (The list displays multiple caches only if you configured a cache cluster.)

3. In the **Search Criteria** section, select the search criteria:

   Note: Because Oracle 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, Oracle 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 Section 6.5.1 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 Section 7.6 and Section 7.9.
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. Oracle 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, Oracle Web Cache lists the URLs for only those objects that meet the criteria.

      If more objects than the number specified meet the invalidation criteria, Oracle 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 additional objects, send another preview request with a different **From** number that specifies the start of the next set of objects.
   c. Click **Submit.**

      Oracle Web Cache displays the Invalidation Preview Results message box, which lists the objects that meet the invalidation criteria. Oracle 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 listed objects are for those 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 an option to specify how to process invalid objects:
   - **Remove immediately:** Oracle 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:** Oracle 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.
9. Click **Submit.**

Oracle Web Cache processes the invalidation request, and returns a Cache Cleanup dialog box which shows the invalidation status.

> **Note:** For prefix-based invalidations that require Oracle 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 Oracle Web Cache.

7.7.3 Using 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.

Oracle 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 following directories:

```
(UNIX) ORACLE_HOME/webcache/toolkit
(Windows) ORACLE_HOME\webcache/toolkit
```

For more information about these APIs, see `readme.toolkit.html` in the following directories for further information about the APIs:

```
(UNIX) ORACLE_HOME/webcache/docs
(Windows) ORACLE_HOME\webcache/docs
```

7.7.4 Using 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. Use the `UTL_TCP` Oracle supplied package to send invalidation requests through database triggers.

For more information, see Oracle PL/SQL documentation.

7.7.5 Using 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 Oracle Web Cache. To achieve this result, you can modify the PERL script such that when the bulk loading operation has completed, the script sends 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 Oracle Web Cache enables administrators to use broad brush strokes when invalidating content, making it safe to invalidate all catalog content even if only a fraction of that content has changed.

7.8 Enabling Response-Header Invalidation

Response header invalidation is an Oracle Web Cache feature that enables an origin server to invalidate cached content through an HTTP response header.

The response-header field supports the following syntax:

```
Oracle-WebCache-Invalidate:
([SYNCHRONOUS=ON|OFF,]
(URI="value" | ( URI_DIR="value" [(;S_KEY="value")*] )
             | ( S_KEY="value"   [(;S_KEY="value")*] ))
([,SYNCHRONOUS=ON|OFF,]
  URI="value" | ( URI_DIR="value" [(;S_KEY="value")*] )
              | ( S_KEY="value"   [(;S_KEY="value")*] ))*
[,SYNCHRONOUS=ON|OFF]
```

Table 7–5 describes the control directives for response header invalidation.

Table 7–5 Control Directives for Oracle-WebCache-Invalidate

Control Directive	Description
SYNCHRONOUS	The SYNCHRONOUS directive enables Oracle Web Cache to determine whether to complete the invalidation before returning the response to the client. It applies to the invalidation as a whole—the combination of all invalidation response headers for a given response. By default, Oracle Web Cache waits for the invalidation to complete (SYNCHRONOUS=ON) before returning the response. Typically the original request updates content hosted by the origin server, and the origin server, in turn, ensures that Oracle Web Cache invalidates all its entries associated with the content before the client receives a response. There is a direct link between the original request and the content identified in the invalidation response header. If an origin server appends an invalidation response header to a random request, the client sending the request should not have to wait for the invalidation to complete. In this case, the origin server should direct Oracle Web Cache to return the response before proceeding with the invalidation (SYNCHRONOUS=OFF).
URI	An invalidation specification with the URI option directive enables Oracle Web Cache to invalidate the entry with the specified URI; this corresponds to basic invalidation.
URI_DIR	An invalidation specification with the URI_DIR option directive enables Oracle Web Cache to interpret the specified URI as a directory and to invalidate all entries stored in the specified directory; this corresponds to URI prefix invalidation, a small but often used subset of advanced invalidation. Note that the directory URI strings must end in a slash (/) to make the URI_DIR option directive consistent with current URI prefix invalidation.
S_KEY	An invalidation specification with the S_KEY option directive enables Oracle Web Cache to interpret the quoted string as a search key; this corresponds to search key invalidation, another small subset of advanced invalidation. Search key matching is case sensitive, the same as it is for traditional invalidation. When a S_KEY option directive appears without an explicit URI directory, Oracle Web Cache uses an implicit URI directory equivalent to the root of the site definition associated with the incoming request. In particular, if the site definition contains a path prefix, the implicit URI directory includes this path prefix.
Conjoined Multiple Directives	An invalidation response header may contain a URI directory followed by one or more search keys. In this situation, a semicolon (;) delimiter separates each directive. When this occurs, a Oracle Web Cache entry must match all the directives to qualify for invalidation.
Multiple Invalidation Directives	When an invalidation response header contains multiple invalidation directives with each consecutive pair of invalidation directives separated by a comma, an Oracle Web Cache entry must match at least one invalidation directive to qualify for invalidation. In other words, Oracle Web Cache treats each comma-delimited invalidation directive as an independent invalidation operation.
Mixing Commas and Semicolons	When an invalidation response header contains both kinds of separators, commas and semicolons, semicolons take precedence. In other words, the consecutive directives separated by semicolons must be examined; then consecutive directives separated by commas are examined.
Multiple Invalidation Response Headers	An origin server can store multiple invalidation response headers in its response to Oracle Web Cache. When this happens, an Oracle Web Cache entry only needs to match one header to qualify for invalidation. In other words, the content of multiple invalidation response headers in the same response are treated as if they were part of a single response header joined by commas. If a response contains at least one invalid invalidation response header, no invalidation takes place even if the response contains other valid invalidation response headers.

Usage Notes

- An invalidation response header consists of the header name "Oracle-WebCache-Invalidate" followed by a colon (:), followed by one or more invalidation directives with consecutive pairs of invalidation directives

separated by a comma (,). Optional synchronicity directives may appear before or after any directive.

- A synchronicity directive consists of the keyword "SYNCHRONOUS" followed by an equal sign (=) followed by either the keyword "ON"or the keyword "OFF".
- An invalidation consists of either a URI or a multi-directive specification.
- A URI directive consists of a URI option directive followed by an equal sign (=) followed by a quoted string.
- A URI option directive consists of the keyword URI.
- A multi-directive specification has two distinct formats:
  - an explicit directory
  - an implicit directory
- The explicit directory format consists of a URI directory directive followed by zero or more search key directives with consecutive search keys separated by a semicolon (;) delimiter.
- The implicit directory format consists of 1 or more search keys with consecutive search keys separated by a semicolon (;) delimiter.
- A URI directory consists of a URI directory option directive followed by an equal sign (=) followed by a quoted string.
- A URI directory option directive consists of the keyword URI_DIR.
- A search key consists of a search key option directive followed by an equal sign (=) followed by a quoted string.
- A search key option directive consists of the keyword S_KEY.
- A quoted string contains either a URI or a search key. Case-sensitivity rules and allowable character sets for URI and search key strings are the same as for other invalidation functionality.
- For fully qualified URIs, a valid scheme includes either http:// or https://, and a valid host name (for example, www.host1.com).
- Port numbers, when specified, must be valid as well. When a URI does not contain a port number, Oracle Web Cache assumes a default port number of 80 for HTTP and 443 for HTTPS. For implicit URI directories, Oracle Web Cache determines the directory based on the site of the original request. Oracle Web Cache ensures that the resulting site definition matches the site definition associated with the original request. In other words, as a security precaution, Oracle Web Cache disallows cross-site invalidation.
- Note that with site-to-server mappings involving wildcards, some requests may have no associated site definition. In this case, the principal of conservatism applies, and Oracle Web Cache disallows the response header invalidation.

7.8.1 Example Usage

The following sections provide examples of invalidation response headers.

The examples are based on the fictional Web application for Harry's Hardware store with the Web site:

```
http://www.HarrysHardware.com
```

At this site, Harry publishes descriptions (including the retail price) for all the popular items he sells. Harry improves the response time for his on-line customers by deploying a Oracle Web Cache in front of the Web server hosting his site, but to ensure his on-line customers see fresh content, he wants the application to invalidate relevant Oracle Web Cache entries whenever it updates descriptions or prices for items in the store. The examples below indicate how to use response header invalidation for various scenarios that Harry has identified.

The sections include:

7.8.1.1 Basic URI Invalidation

Harry sells one particularly popular item called the Thor hammer, and the corresponding description page has the following URI:

```
http://www.harryshardware.com/products/tools/hammers/Thor.html
```

If Harry decides to put the Thor on sale, his Web application could invalidate the appropriate Web Cache entry with an invalidation response header containing a fully specified URI:

```
Oracle-WebCache-Invalidate:
URI="http://www.harryshardware.com/products/tools/hammers/Thor.html"
```

If the original request specified the host name (`www.harryshardware.com`) explicitly, the application could return an invalidation response header with a path-only URI:

```
Oracle-WebCache-Invalidate: URI="/products/tools/hammers/Thor.html"
```

7.8.1.2 Directory URI Invalidation

If Harry decides to put all the hammers in his store on sale, his Web application could invalidate all Web Cache entries for hammers with an invalidation response header containing a fully specified URI directory:

```
Oracle-WebCache-Invalidate: URI_
DIR="http://www.harryshardware.com/products/tools/hammers/"
```

If the original request specified the host name (`www.harryshardware.com`) explicitly, the application could return an invalidation response header with a path-only URI directory:

```
Oracle-WebCache-Invalidate: URI_DIR="/products/tools/hammers/"
```

7.8.1.3 Asychronous Invalidation

In the examples so far, we have not specified a synchronicity directive, so by default Oracle Web Cache would complete the invalidation before returning the response to the client.

If Harry wanted the example from Section 7.8.1.2 to proceed asynchronously, that is, if he did not want Oracle Web Cache to wait for the invalidation to complete before returning the response, his Web application could send an invalidation response header that looks like this:

```
Oracle-WebCache-Invalidate: SYNCHRONOUS=OFF,
URI_DIR="http://www.harryshardware.com/products/tools/hammers/"
```

Notice that the response header above contained a fully qualified URI directory. If the original request specified the host name (`www.harryshardware.com`) explicitly, the application could return an invalidation response header with a path-only URI directory:

```
Oracle-WebCache-Invalidate: SYNCHRONOUS=OFF, URI_DIR="/products/tools/hammers/"
```

7.8.1.4 Search Key Invalidation with Explicit URI

Suppose that Harry wants to reduce the price of all TrueSaw saws but not the handsaws, just the power saws (for example, skill saws and chainsaws). His Web application could invalidate all the necessary entries with an invalidation response header that looks like this:

```
Oracle-WebCache-Invalidate: URI_
DIR="http://www.harryshardware.com/products/tools/saws/";S_KEY="PowerTool";
S_KEY="TrueSaw"
```

Notice the addition of the `S_KEY` directives to ensure the invalidation of only TrueSaw power saws.

Remember, when an invalidation response header contains multiple directives separated by semicolons, an Oracle Web Cache entry must match all directives for the invalidation to take place.

Notice also that the response header above contained a fully qualified URI directory. If the original request specified the host name (`www.harryshardware.com`) explicitly, the application could return an invalidation response header with a path-only URI directory:

```
Oracle-WebCache-Invalidate: URI_DIR="/products/tools/saws/";S_KEY="PowerTool";S_
KEY="TrueSaw"
```

7.8.1.5 Search Key Invalidation with Implicit URI

Suppose that Harry sets up an additional site for selling large appliances (dishwashers, refrigerators, etc.). Suppose also that he defines this site using a path prefix of `/products/appliances`. The following Table 7–6 are the site definitions for the Web site:

Table 7–6 Web Site Definitions

Scheme	Host	Port Number	Path Prefix
`http`	`www.harryshardware.com`	`80`	`/products/appliances`

Table 7–6 (Cont.) Web Site Definitions

Scheme	Host	Port Number	Path Prefix
http	www.harryshardware.com	80	/

The first site pertains strictly to large appliances; the second site applies to everything else in Harry's store.

Suppose further that Harry changes the price for all KeepCold refrigerators and that the site definition for an incoming request pertains to Harry's appliance site; scheme http, host name `www.harryshardware.com`, (optional) port 80 and path prefix of `/products/appliances`. His Web application could invalidate all the necessary entries with an invalidation response header that looks like this:

```
S_KEY="KeepCold";S_KEY="Refrigerators"
```

Notice that the invalidation response header contains only search key directives; it does not contain a URI directory directive. When this happens, Oracle Web Cache forms an implicit URI directory from the site definition associated with the incoming request. In this case the implicit directory corresponds to:

```
http://www.harryshardware.com/products/appliances/
```

As before, with multiple directives separated by semicolons, an Oracle Web Cache entry must match all directives for the invalidation to take place.

The equivalent invalidation response header with an explicit, fully qualified URI directory would look like this:

```
Oracle-WebCache-Invalidate: URI_
DIR="http://www.harryshardware.com/products/appliances/";S_KEY="KeepCold";
S_KEY="Refrigerators"
```

The equivalent invalidation response header with an explicit, path-only URI directory would look like this:

```
Oracle-WebCache-Invalidate: URI_DIR="/products/appliances/";S_KEY="PowerTool";
S_KEY="TrueSaw"
```

7.8.1.6 Multiple Invalidation Directives

Suppose that Harry wants to upgrade his entire inventory of drills and wrenches. His Web application could invalidate all the necessary entries with a response containing the following invalidation response header:

```
Oracle-WebCache-Invalidate: URI_
DIR="http://www.harryshardware.com/products/tools/drills/",
URI_DIR="http://www.harryshardware.com/products/tools/wrenches/"
```

Remember, when an invalidation response header contains two consecutive invalidation specifications separated by a comma, an Oracle Web Cache entry only needs to match one invalidation specification for the invalidation to take place.

Notice that the response header above contained fully qualified URI directories. If the original request specified the host name (`www.harryshardware.com`) explicitly, the application could return an invalidation response header with path-only URI directories:

```
Oracle-WebCache-Invalidate: URI_DIR="/products/tools/drills/", URI_
```

```
DIR="/products/tools/wrenches/"
```

7.8.1.7 Mixing Commas and Semicolons

Suppose that Harry wants to put both the Thor hammer and all TrueSaw power saws on sale. His Web application could invalidate all the necessary entries with a response containing the following invalidation response header:

```
Oracle-WebCache-Invalidate:
URI="http://www.harryshardware.com/products/tools/hammers/Thor.html",
URI_DIR="http://www.harryshardware.com/products/tools/saws/";S_KEY="PowerTools";S_
KEY="TrueSaw"
```

Notice the use of both the comma and semicolon as separators. In this instance, the first directive consists of only the URI for the Thor hammer. The second directive consists of three invalidation specifications: the URI directory for saws and the search keys for Power Tools and TrueSaw tools. Semicolons take precedence over commas.

Notice also that the response header above contained a fully qualified URI and a fully qualified URI directory.

If the original request specified the host name (`www.harryshardware.com`) explicitly, the application could return an invalidation response header with a path-only URI and a path-only URI directory:

```
Oracle-WebCache-Invalidate: URI="/products/tools/hammers/Thor.html"
URI_DIR="/products/tools/saws/";S_KEY="PowerTools";S_KEY="TrueSaw"
```

7.8.1.8 Multiple Invalidation Response Headers

Returning to the example of Section 7.8.1.6, a Web application, alternatively, could invalidate all the necessary entries with a response containing two separate invalidation response headers:

```
Oracle-WebCache-Invalidate: URI_
DIR="http://www.harryshardware.com/products/tools/drills/"
Oracle-WebCache-Invalidate: URI_
DIR="http://www.harryshardware.com/products/tools/wrenches/"
```

The directives from two different invalidation response headers in the same response are treated as if they were separate directives in a single response header—that is, they are treated as if they were separated by commas in a single invalidation response header. Notice, too, the response headers contained a fully qualified URI directory. If the original request specified the host name (`www.harryshardware.com`) explicitly, the application could return invalidation response headers with path-only URI directories:

```
Oracle-WebCache-Invalidate: URI_DIR="/products/tools/drills/"
Oracle-WebCache-Invalidate: URI_DIR="/products/tools/wrenches/"
```

7.9 Enabling Search Keys for Invalidations

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 Section 7.6 for a complete description of the `Surrogate-Key` response-header field.

   By default, Oracle 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 these methods:

- For out-of-band invalidations:
  - Use the **Search Keys** section from the Advanced Content Invalidation page (**Operations** > **Advanced Content Invalidation**) of Oracle Web Cache Manager. See Section 7.7.2 for instructions on using Oracle 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 Section 7.5.1.
- For ESI inline invalidations, specify the `<esi:invalidate>` tag with the `OTHER` element to use the `ADVANCEDSELECTOR` element, and specify the `NAME` and `VALUE` attributes to use `SEARCHKEY` and the search key value, respectively. See Section 11.4.6.
- For response header invalidation, specify the `S_KEY` option directive. See Section 7.8.

7.10 Security Considerations

This section covers the following topics:

- Section 7.10.1, "About the invalidator account"
- Section 7.10.2, "Propagation of Invalidation Messages"

7.10.1 About the invalidator account

To invalidate objects in the cache, send an HTTP POST request from the `invalidator` account through an invalidation listening port.

The `invalidator` account is an administrator authorized to send invalidation requests. See Section 5.2 for further information about configuring password security.

7.10.2 Propagation of Invalidation Messages

Propagation of invalidation messages from one Oracle Web Cache server to another occurs in the following deployments:

- Cache cluster with multiple Oracle Web Cache servers
- Cache hierarchy whereby one Oracle Web Cache server acts as an origin server to another Oracle Web Cache server

7.10.2.1 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 Oracle 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, Oracle 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 continue to respond to requests. However, because other cluster members have a different version of the configuration, the caches do 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, Oracle Web Cache forwards the request to the origin server.

When the cache cluster members are not running the same version of Oracle 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 Oracle Web Cache.

See the *Oracle Fusion Middleware Upgrade Planning Guide* for more information about upgrading Oracle Web Cache to 11*g* Release 1 (11.1.1), including information about upgrading cache cluster members

7.10.2.2 Invalidation in Hierarchies

In a configuration with a hierarchy of Oracle Web Cache servers, a **cache hierarchy**, it is likely that content is cached on multiple servers.

Figure 7–2 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.

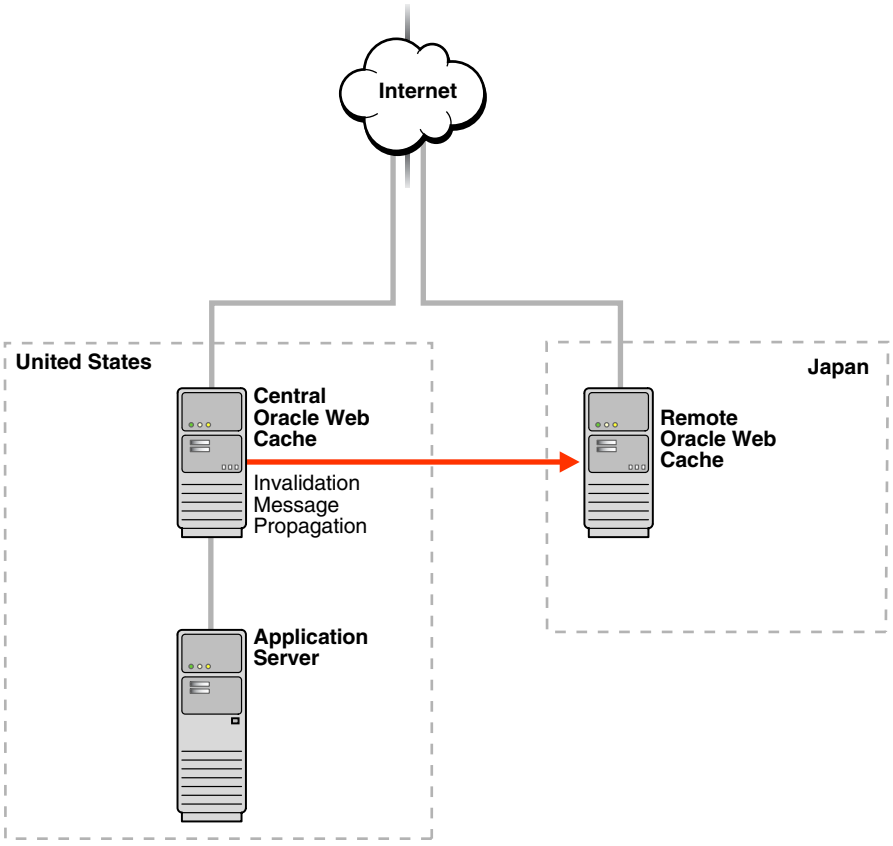
The **central cache** uses the `invalidator` account name and password of the remote or subscriber Oracle Web Cache server. The invalidation request specifies the objects to invalidate, as well as the site host name of the objects. The site host name is compared with the IP address of the cache from which the invalidation request was propagated. If there is a match, the cache processes the invalidation request. Otherwise, the request is rejected.

For automatic propagation of invalidation messages, Oracle Web Cache passes the encoded `invalidator` password in the page request between the central and **remote 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:

- Disable the default HTTP port and configure an HTTPS port in its place for the central cache. See Section 5.4.2.
- Disable the default HTTP port for invalidation and configure an HTTPS port in its place for the remote cache. See Section 5.5.1.

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 7–2 Invalidating Content in a Distributed Cache Hierarchy



To ensure that the central cache only invalidates its content, the remote cache checks the site host name specified in the invalidation message with the IP address of the central cache from which the invalidation message was propagated. If there is a match, the remote cache processes the invalidation request. Otherwise, the request is rejected. The site host name for the central and remote caches should be configured to be identical, making a mismatch unlikely.

See Section 10.2 for instructions on configuring a cache hierarchy.

8

Using Diagnostic Features

This chapter describes the diagnostic features available with Oracle Web Cache.

This chapter includes the following topics:

8.1 Introduction to Diagnostic Solutions

Oracle Web Cache provides a number diagnostic tools to enable to you evaluate performance and optimal configuration settings. These tools include:

- Popular requests reporting. See Section 8.2.
- Varying levels of statistics monitoring through Fusion Middleware Control. See Section 8.4 and Section 8.5.
- Displaying diagnostics information in the `Server` response-header field or as a textual string in the HTML response body of an object. See Section 8.3.

8.2 Introduction to Listing Popular Requests and Cache Contents

You can view a list of the most popular requests and a list of the contents of the cache, as well as requests that were not 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. See Section 8.6.

- A list of the requested URLs for objects not stored in the cache.

You can use this list to verify that the caching rules are caching the correct objects. See Section 8.6.

- A list of the cached or non-cached URLs of the objects to an exported file

  Oracle Web Cache can write a list of content of the cache to a file. See Section 8.7.

8.3 Introduction to Displaying Diagnostic and Event Log Information in the HTML Body or Server Response-Header Field

By default, Oracle 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. When 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: text/html` response-header field. When enabled, the diagnostic information appears within HTML comment tags:

`<!-- Web Cache Debug Info: diagnostic_information-->`

For objects sent to browsers, Oracle Web Cache adds diagnostic information to the `Server` response-header field of the HTTP response message:

`Server: Oracle Fusion Middleware 11g (multiple_version_object_version_number) Server_header_from_origin_server Oracle-Web-Cache-11g/11.1.1.0.0 (diagnostic_information)`

The `Server` response-header field specifies name/value pairs for Oracle HTTP Server and Oracle Web Cache. The information for Oracle 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 8–1 describes the diagnostic fields.

Table 8–1 Control Directives for Server

Control Directive	Description
ESI_processing_type	*ESI_processing_type* is: ■ 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* is: ■ 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:

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 Fusion Middleware 11g (11.1.1.0.0) Oracle-HTTP-Server
Oracle-Web-Cache-11g/11.1.1.0.0 (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.

For more information, see:

- Section 9.4 for further information about creating a user-defined access log format that includes the sc(Server) field
- Section 8.8 for configuring the display of diagnostics information

8.4 Viewing General and Detailed Statistics

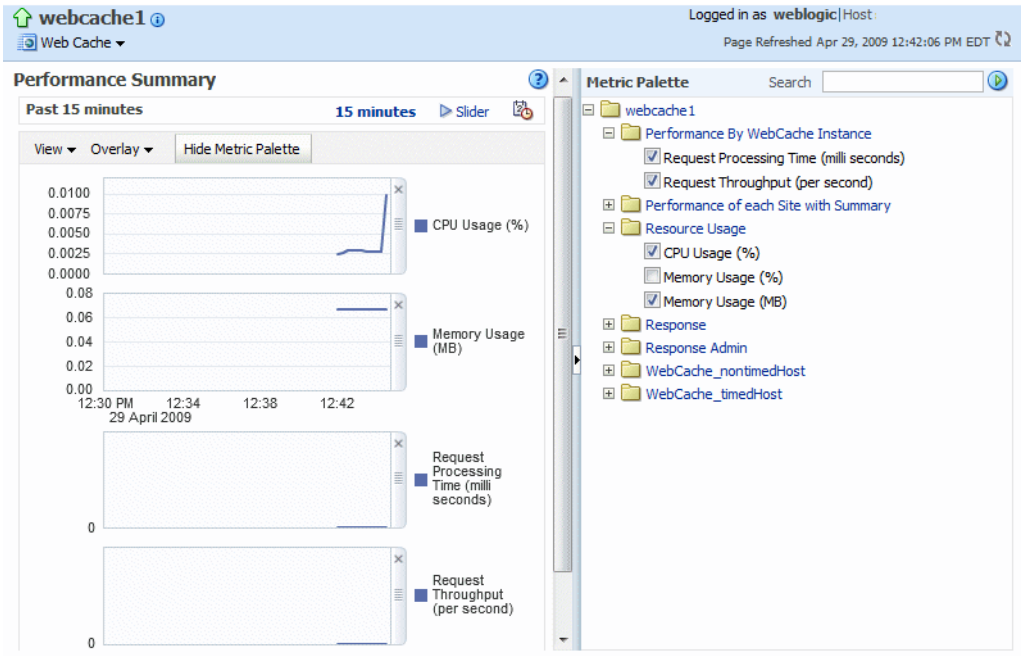
To view general statistics, navigate to the Web Cache Home page. See Section 2.6.3 for further information about the statistics provided in the Web Cache Home page.

You can also view detailed statistics about a specific cache instance:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Monitoring** and then **Performance Summary**.

   The Performance Summary page displays with performance metrics.
3. To see additional metrics, click **Show Metric Palette** and expand the metric categories.

   The following figure show the Performance Summary page with the Metric Palette displayed:



4. Select additional metrics and add them to the Performance Summary.

   To obtain a definition of a performance metric, and information about the actions you should take when the metric is out of range, right-click the name of the metric and select **Help** from the context menu.

If after monitoring metrics, you need additional performance metrics, point your browser to the following URL:

`http://web_cache_hostname:stat_port`

This URL takes you to the Oracle Web Cache Internal Diagnosability Monitor page, which provides additional information about cache hits and misses. See Section 2.11.1 for further information about determining the statistics monitoring port.

8.5 Viewing Configuration Statistics

Table 8–2 explains where to find information for evaluating the effectiveness of configuring rules.

Table 8–2 Configuration Statistics

Type of Statistic	See Also
Request filters and rules	Section 4.13
Caching rules	Section 6.9

8.6 Listing Popular Requests

To understand how evaluating the most popular requests can help determine if the caching rules are caching the correct objects, see Section 8.2.

To view the list of URLs of the most popular requests:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Monitoring** and then **Popular Requests**.

   The Popular Request page displays.
3. From the **Show Popular Requests** list, select an option:
   - **All**: Select to display all requests received by the cache.
   - **Cache Popular Requests**: Select to display only those requests stored in the cache.
   - **Non Cache Popular Requests**: Select to display only those requests not stored in the cache.
4. In **Number of Popular Requests**, enter the maximum number of URLs you want displayed.
5. Click **Go.**

   The table updates with the list of URLs of requests since the cache was last started. The table contains the following columns for each request:
   - **Site:** This column displays the requested site name.
   - **HTTP Method:** This column displays if the request was using a GET, GET with query string, or POST HTTP request method.
   - **URL:** The column displays the URL of the object. The URLs may contain additional descriptive information, such as cookie or session information.
   - **POST Body:** This column displays the POST body contents.
   - **Cached?:** This column display whether 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, Oracle Web Cache serves some stale objects until the origin servers have the capacity to refresh them.)
     - **Yes, but needs validation:** The object is cached, but it requires validation by the origin server to be served out of the cache. Validation is done through a simple conditional request from Oracle Web Cache to the origin server using the cached object's unique validator.
     - **No:** The object is not cached.

- **Caching Reason**: The reason that the object is cached or not cached. Possible values are:
  - **ACL document:** Cached or not cached because the object is an Access Control List (ACL) document for authorizing the access of a user to ACL-protected pages.
  - **By caching <rule>:** Cached or not cached because of a caching rule.
  - **By ETag response header:** Cached or not cached because of the ETag response header.
  - **By HTTP headers:** Cached or not cached because of information in the HTTP header.
  - **By HTTP response code:** Cached or not cached because of the HTTP response code. Normally any response code that is not 200 is not cached, but some non-200 responses can get cached because of a caching rule specifically allowing for it.
  - **By reference TTL in ESI tag:** Cached or not cached because of the nonzero value of the reference TTL (time-to-live parameters) specified in the ESI tag.
  - **By Surrogate-Control response header:** Cached or not cached because of information in the Surrogate-Control response header.
  - **By X-Oracle-Cache response header:** Cached or not cached because of information in the `X-Oracle-Cache` response header.
  - **Cookie mismatch:** Cached or 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.
  - **No directive or rule:** Cached or not cached because no directive or rule has stated that the object should be cached.
  - **Not a GET or POST method:** Not cached because the object was not a GET or POST method.
  - **Object is too large:** Not cached because the object is larger than the size specified as the **Maximum Size of Single Cached Object** specified in the Resource Limits and Timeout page.
  - **POST body too large:** Cached or not cached because the POST body was too large to be cached.
  - **URL contains query string:** Cached or not cached because the request contains a query string but the request did not match any caching rules.
- **Size**: The column displays the size of the requested object. The size is represented in bytes, kilobytes (KB), or megabytes (MB).
- **Compressed?:** This column displays the reason the object was compressed or not compressed:
  - **Yes, by caching rule:** Compressed because the object matched a caching rule that enabled or disabled compression.
  - **Yes, by MIME type:** Compressed because the object's MIME content type.
  - **No, by default setting:** Compression is enabled for the site and the browser accepts GZIP compressed response, but there is no matching caching rule and the response does not contain a `compress` control header in the `Surrogate-Control` response header or a MIME type.

- **Yes, by Surrogate-Control header:** Compressed or not compressed because of the setting of the `compress` control directive in the `Surrogate-Control` response header.
- **Yes, by caching rule:** Compressed or not compressed because the object matched a caching rule that enabled or disabled compression.
- **No, by MIME type:** Not compressed because the object's MIME content type.
- **No, by default setting:** Compression is enabled for the site and the browser accepts GZIP compressed responses, but there is no matching caching rule and the response does not contain a `compress` control header in the `Surrogate-Control` response header or a MIME type. See Section 1.2.5 to better understand when Oracle Web Cache automatically disables compression.
- **No, by Surrogate-Control header:** Not compressed because of the setting of the `compress` control directive in the `Surrogate-Control` response header.
- **No, limited browser support:** Not compressed because the client's browser has bugs and cannot handle receiving compressed objects.
- **No, needs Web Cache processing:** Not compressed because the object requires parsing and tag process. For example, objects containing ESI tag requiring processing before there can be any cache hits.
- **No, browser capability:** Not compressed because the client's browser did not indicate to Oracle Web Cache that it could accept GZIP compressed responses. Therefore, Oracle Web Cache does not compress any responses sent to this browser.
- **No, disabled for site:** Not compressed because compression was disabled for the entire site. Section 2.11.3 to enable compression for a site.
- **No, object too small:** Not compressed because the object was less than 23 bytes for compression to be beneficial.
- **No, routing only mode:** Not compressed because the `ROUTINGONLY` attribute is set to `YES` in the `webcache.xml` file. See Section 3.8 for further information about this attribute.

8.7 Listing Cache Contents to a File

To generate a list of the URLs of all of the objects currently stored in the cache to a file named `webache_contents.txt`:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Monitoring** and then **Popular Requests**.

   The Popular Request page displays.
3. From the **Filter Popular Request By** list, select an option:
   - **All**: Select to display all requests received by the cache.
   - **Cache Popular Requests**: Select to display only those requests stored in the cache.

- **Non Cache Popular Requests**: Select to display only those requests not stored in the cache.

4. In **Number of Popular Requests**, enter the maximum number of URLs you want displayed.
5. Click **Export File**.
6. In the message dialog, click **OK** to export the contents.

   Oracle Web Cache writes the list of URLs to `webcache_contents.txt` in this directories:

   (UNIX) *ORACLE_INSTANCE*/diagnostics/logs/WebCache/<*webcache_name*>
   (Windows) *ORACLE_INSTANCE*\diagnostics\logs\WebCache\<*webcache_name*>

Each time you generate the list, Oracle 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 Oct 20 11:47:03 2008
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 Oct 25 13:01:24 2008
www.company.com:80/images/white_spacer_xp.gif
www.company.com:80/images/white_spacer.gif
www.company.com:80/images/miniappsnet.gif
.
.
.
```

8.8 Configuring Where to Display Diagnostic Information

To understand how Oracle Web Cache can add diagnostic information to the `Server` response-header field or as a textual string in the HTML response body of an object, see Section 8.3.

To configure diagnostic information in the `Server` response-header field or the HTML response body:

1. From Oracle Web Cache Manager, in the navigator frame, select **Logging and Diagnostics** > **Diagnostics**. See Section 2.7.2.
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. From the **Global Page Body Diagnostics Configuration** table, 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.**

5. Click **Apply Changes**.

9 Logging

The logging feature of Oracle Web Cache enables you to troubleshoot difficulties you might have in execution and use of Oracle Web Cache and associated processes.

This chapter includes the following topics:

9.1 Introduction to Event Logs

Oracle Web Cache records event and error information in event logs. An event log entry can help you determine what objects have been inserted in the cache and alert you to any cache-related issues. By default, Oracle Web Cache collects all event log messages associated with each request in memory. If the most severe message in the request is at or above the selected verbosity level, Oracle Web Cache writes all the messages related to the request to the event log at once. Oracle Web Cache groups the messages for the request together in the log file for easier diagnosis.

By default, the event log has a file name of `event_log` for the Oracle Web Cache and Oracle Diagnostic Logging (ODL) text formats and `log.xml` for the ODL XML format. Oracle Web Cache stores logs files in the following directories:

```
(UNIX) ORACLE_INSTANCE/diagnostics/logs/WebCache/<webcache_name>
(Windows) ORACLE_INSTANCE\diagnostics\logs\WebCache\<webcache_name>
```

This section includes the following topics:

9.1.1 Event Logging Formats

When you configure settings for event logs, select the logging format:

9.1.1.1 Oracle Diagnostics Logging Text and XML Formats

The Oracle Diagnostic Logging (ODL) format provides a common format for all diagnostic messages and log files, and a mechanism for correlating the diagnostic messages from various components across Oracle Fusion Middleware.

You can select ODL Text to create a text file or ODL XML to create an XML file.

The format of the ODL Text format follows:

```
[TSTZ_ORIGINATING] [MSG_TYPE:MSG_ID] [MODULE_ID;MSG_LEVEL] [MODULE_ID] [ECID] MSG_
TEXT
```

Table 9–4 describes the fields for the ODL Text format.

Table 9–1 ODL Text Message Fields

Fields	Description
TSTZ_ORIGINATING	The date and time when the message was generated. Time is either displayed in local or Greenwich Mean Time.
MSG_TYPE	The type of message. Possible values are NOTIFICATION, WARNING, TRACE, and DEBUG.
MSG_LEVEL	The message level, represented by an integer value that qualifies the message type. Possible values are from 1 (highest severity) through 32 (lowest severity).
MSG_ID	The ID that uniquely identifies the message within the component. The ID consists of a prefix that represents the component, followed by a dash, then a 5-digit number. For example: WXE-08513. The *Oracle Fusion Middleware Error Messages Reference* describes the messages in further detail.
MODULE_ID	The ID of the module that originated the message. If the component is a single module, the component ID is listed for this attribute.
ECID	The Execution Context ID (ECID), which is a global unique identifier of the execution of a particular request in which the originating component participates. You can use the ECID to correlate error messages from different components. **See Also:** Section 9.1.1.4 for more information about the Oracle-ECID request header
MSG_TEXT	The text of the error message.

The following shows an event log excerpt with the ODL Text format:

```
[2008-11-04T05:55:35-05:00] [webcache] [NOTIFICATION:1] [WXE-08513] [logging]
[ecid: ] Cache server process ID 11679 is starting up.
[2008-11-04T05:55:35-05:00] [webcache] [NOTIFICATION:1] [WXE-09612] [main] [ecid:
] Oracle Web Cache 11g (11.1.1)
[2008-11-04T05:55:35-05:00] [webcache] [NOTIFICATION:1] [WXE-13002] [config]
[ecid: ] Maximum allowed incoming connections are 700
[2008-11-04T05:55:35-05:00] [webcache] [NOTIFICATION:1] [WXE-09446] [stats] [ecid:
] Statistics initialization commencing.
[2008-11-04T05:55:35-05:00] [webcache] [NOTIFICATION:1] [WXE-09441] [stats] [ecid:
] DMS enabled
```

```
[2008-11-04T05:55:35-05:00] [webcache] [NOTIFICATION:1] [WXE-09447] [stats] [ecid:
] Statistics initialization complete.
[2008-11-04T05:55:36-05:00] [webcache] [NOTIFICATION:1] [WXE-12209] [cluster]
[ecid: ] A 1 node cluster successfully initialized
[2008-11-04T05:55:36-05:00] [webcache] [NOTIFICATION:1] [WXE-09614] [main] [ecid:
] The following Oracle Web Cache internal files are pre-populated to the cache:
[[/host:port/_oracle_http_server_webcache_static_.html]]
```

Table 9–2 describes the fields for the ODL XML format.

Table 9–2 ODL XML Message Fields

Fields	Description
TSTZ_ORIGINATING	The date and time when the message was generated. Time is either displayed in local or Greenwich Mean Time.
COMPONENT_ID	The ID of the component that originated the message.
MSG_ID	The ID that uniquely identifies the message within the component. The ID consists of a prefix that represents the component, followed by a dash, then a 5-digit number. For example: WXE-08513.
MSG_TYPE	The type of message. Possible values are NOTIFICATION, WARNING, TRACE, and DEBUG.
MSG_LEVEL	The message level, represented by an integer value that qualifies the message type. Possible values are from 1 (highest severity) through 32 (lowest severity).
HOST_ID	The name of the host where the message originated.
HOST_NWADDR	The network address of the host where the message originated.
MODULE_ID	The ID of the module that originated the message. If the component is a single module, the component ID is listed for this attribute.
ECID	The Execution Context ID (ECID), which is a global unique identifier of the execution of a particular request in which the originating component participates. You can use the ECID to correlate error messages from different components. **See Also:** Section 9.1.1.4 for more information about the Oracle-ECID request header
MSG_TEXT	The text of the error message.

The ODL XML Format provides additional fields, such as the following shows an event log excerpt for the ODL XML format:

```
<MESSAGE>
  <HEADER>
    <TSTZ_ORIGINATING>2008-11-04T06:07:14.0116-05:00</TSTZ_ORIGINATING>
    <COMPONENT_ID>WXE</COMPONENT_ID>
    <MSG_ID>8513</MSG_ID>
    <MSG_TYPE TYPE="NOTIFICATION"></MSG_TYPE>
    <MSG_LEVEL>1</MSG_LEVEL>
    <HOST_ID>host</HOST_ID>
    <HOST_NWADDR>10.10.150.35</HOST_NWADDR>
    <MODULE_ID>logging</MODULE_ID>
  </HEADER>
  <CORRELATION_DATA>
    <EXEC_CONTEXT_ID>
```

```
      <UNIQUE_ID>-</UNIQUE_ID>
      <SEQ>0</SEQ>
    </EXEC_CONTEXT_ID>
  </CORRELATION_DATA>
  <PAYLOAD>
    <MSG_TEXT>Cache server process ID 13176 is starting up.
</MSG_TEXT>
  </PAYLOAD>
</MESSAGE>
<MESSAGE>
  <HEADER>
    <TSTZ_ORIGINATING>2008-11-04T06:07:14.0117-05:00</TSTZ_ORIGINATING>
    <COMPONENT_ID>WXE</COMPONENT_ID>
    <MSG_ID>9612</MSG_ID>
    <MSG_TYPE TYPE="NOTIFICATION"></MSG_TYPE>
    <MSG_LEVEL>1</MSG_LEVEL>
    <HOST_ID>host</HOST_ID>
    <HOST_NWADDR>10.10.150.35</HOST_NWADDR>
    <MODULE_ID>main</MODULE_ID>
  </HEADER>
  <CORRELATION_DATA>
    <EXEC_CONTEXT_ID>
      <UNIQUE_ID>-</UNIQUE_ID>
      <SEQ>0</SEQ>
    </EXEC_CONTEXT_ID>
  </CORRELATION_DATA>
  <PAYLOAD>
    <MSG_TEXT>Oracle Web Cache 11g (11.1.1)
</MSG_TEXT>
  </PAYLOAD>
</MESSAGE>
<MESSAGE>
  <HEADER>
    <TSTZ_ORIGINATING>2008-11-04T06:07:14.0118-05:00</TSTZ_ORIGINATING>
    <COMPONENT_ID>WXE</COMPONENT_ID>
    <MSG_ID>13002</MSG_ID>
    <MSG_TYPE TYPE="NOTIFICATION"></MSG_TYPE>
    <MSG_LEVEL>1</MSG_LEVEL>
    <HOST_ID>host</HOST_ID>
    <HOST_NWADDR>10.10.150.35</HOST_NWADDR>
    <MODULE_ID>config</MODULE_ID>
  </HEADER>
  <CORRELATION_DATA>
    <EXEC_CONTEXT_ID>
      <UNIQUE_ID>-</UNIQUE_ID>
      <SEQ>0</SEQ>
    </EXEC_CONTEXT_ID>
  </CORRELATION_DATA>
  <PAYLOAD>
    <MSG_TEXT>Maximum allowed incoming connections are 700
</MSG_TEXT>
  </PAYLOAD>
</MESSAGE>
<MESSAGE>
  <HEADER>
    <TSTZ_ORIGINATING>2008-11-04T06:07:14.0191-05:00</TSTZ_ORIGINATING>
    <COMPONENT_ID>WXE</COMPONENT_ID>
    <MSG_ID>9446</MSG_ID>
    <MSG_TYPE TYPE="NOTIFICATION"></MSG_TYPE>
    <MSG_LEVEL>1</MSG_LEVEL>
```

```
    <HOST_ID>host</HOST_ID>
    <HOST_NWADDR>10.10.150.35</HOST_NWADDR>
    <MODULE_ID>stats</MODULE_ID>
  </HEADER>
  <CORRELATION_DATA>
    <EXEC_CONTEXT_ID>
      <UNIQUE_ID>-</UNIQUE_ID>
      <SEQ>0</SEQ>
    </EXEC_CONTEXT_ID>
  </CORRELATION_DATA>
  <PAYLOAD>
    <MSG_TEXT>Statistics initialization commencing.
</MSG_TEXT>
  </PAYLOAD>
</MESSAGE>
<MESSAGE>
  <HEADER>
    <TSTZ_ORIGINATING>2008-11-04T06:07:14.0265-05:00</TSTZ_ORIGINATING>
    <COMPONENT_ID>WXE</COMPONENT_ID>
    <MSG_ID>9438</MSG_ID>
    <MSG_TYPE TYPE="NOTIFICATION"></MSG_TYPE>
    <MSG_LEVEL>1</MSG_LEVEL>
    <HOST_ID>host</HOST_ID>
    <HOST_NWADDR>10.10.150.35</HOST_NWADDR>
    <MODULE_ID>stats</MODULE_ID>
  </HEADER>
  <CORRELATION_DATA>
    <EXEC_CONTEXT_ID>
      <UNIQUE_ID>-</UNIQUE_ID>
      <SEQ>0</SEQ>
    </EXEC_CONTEXT_ID>
  </CORRELATION_DATA>
  <PAYLOAD>
    <MSG_TEXT>The statistics persistent repository is being reset by new
configuration
</MSG_TEXT>
  </PAYLOAD>
</MESSAGE>
<MESSAGE>
  <HEADER>
    <TSTZ_ORIGINATING>2008-11-04T06:07:14.1556-05:00</TSTZ_ORIGINATING>
    <COMPONENT_ID>WXE</COMPONENT_ID>
    <MSG_ID>9441</MSG_ID>
    <MSG_TYPE TYPE="NOTIFICATION"></MSG_TYPE>
    <MSG_LEVEL>1</MSG_LEVEL>
    <HOST_ID>host</HOST_ID>
    <HOST_NWADDR>10.10.150.35</HOST_NWADDR>
    <MODULE_ID>stats</MODULE_ID>
  </HEADER>
  <CORRELATION_DATA>
    <EXEC_CONTEXT_ID>
      <UNIQUE_ID>-</UNIQUE_ID>
      <SEQ>0</SEQ>
    </EXEC_CONTEXT_ID>
  </CORRELATION_DATA>
  <PAYLOAD>
    <MSG_TEXT>DMS enabled
</MSG_TEXT>
  </PAYLOAD>
</MESSAGE>
```

```
<MESSAGE>
  <HEADER>
    <TSTZ_ORIGINATING>2008-11-04T06:07:14.1559-05:00</TSTZ_ORIGINATING>
    <COMPONENT_ID>WXE</COMPONENT_ID>
    <MSG_ID>9447</MSG_ID>
    <MSG_TYPE TYPE="NOTIFICATION"></MSG_TYPE>
    <MSG_LEVEL>1</MSG_LEVEL>
    <HOST_ID>host</HOST_ID>
    <HOST_NWADDR>10.10.150.35</HOST_NWADDR>
    <MODULE_ID>stats</MODULE_ID>
  </HEADER>
  <CORRELATION_DATA>
    <EXEC_CONTEXT_ID>
      <UNIQUE_ID>-</UNIQUE_ID>
      <SEQ>0</SEQ>
    </EXEC_CONTEXT_ID>
  </CORRELATION_DATA>
  <PAYLOAD>
    <MSG_TEXT>Statistics initialization complete.
</MSG_TEXT>
  </PAYLOAD>
</MESSAGE>
<MESSAGE>
  <HEADER>
    <TSTZ_ORIGINATING>2008-11-04T06:07:14.5912-05:00</TSTZ_ORIGINATING>
    <COMPONENT_ID>WXE</COMPONENT_ID>
    <MSG_ID>12209</MSG_ID>
    <MSG_TYPE TYPE="NOTIFICATION"></MSG_TYPE>
    <MSG_LEVEL>1</MSG_LEVEL>
    <HOST_ID>host</HOST_ID>
    <HOST_NWADDR>10.10.150.35</HOST_NWADDR>
    <MODULE_ID>cluster</MODULE_ID>
  </HEADER>
  <CORRELATION_DATA>
    <EXEC_CONTEXT_ID>
      <UNIQUE_ID>-</UNIQUE_ID>
      <SEQ>0</SEQ>
    </EXEC_CONTEXT_ID>
  </CORRELATION_DATA>
  <PAYLOAD>
    <MSG_TEXT>A 1 node cluster successfully initialized
</MSG_TEXT>
  </PAYLOAD>
</MESSAGE>
<MESSAGE>
  <HEADER>
    <TSTZ_ORIGINATING>2008-11-04T06:07:20.8036-05:00</TSTZ_ORIGINATING>
    <COMPONENT_ID>WXE</COMPONENT_ID>
    <MSG_ID>9614</MSG_ID>
    <MSG_TYPE TYPE="NOTIFICATION"></MSG_TYPE>
    <MSG_LEVEL>1</MSG_LEVEL>
    <HOST_ID>host</HOST_ID>
    <HOST_NWADDR>10.10.150.35</HOST_NWADDR>
    <MODULE_ID>main</MODULE_ID>
  </HEADER>
  <CORRELATION_DATA>
    <EXEC_CONTEXT_ID>
      <UNIQUE_ID>-</UNIQUE_ID>
      <SEQ>0</SEQ>
    </EXEC_CONTEXT_ID>
```

```
  </CORRELATION_DATA>
  <PAYLOAD>
    <MSG_TEXT>The following Oracle Web Cache internal files are pre-populated to
the cache: [[/host:port/_oracle_http_server_webcache_static_.html]]
</MSG_TEXT>
  </PAYLOAD>
</MESSAGE>
```

For more information about the ODL format, see:

- *Oracle Fusion Middleware Administrator's Guide* for more information about ODL messages and ODL log files
- *Oracle Fusion Middleware Administrator's Guide* for information about configuring the amount of information written to log files.

9.1.1.2 Oracle Web Cache Classic Format

The Oracle Web Cache log format is intended for customers who prefer the traditional log format provided by Oracle Web Cache in previous releases.

The format of the Oracle Web Cache format follows:

```
[TIMESTAMP] [MSG_TYPE MSG_ID] [ECID] MSG_TEXT
```

Table 9–3 describes the fields for Oracle Web Cache format.

Table 9–3 Oracle Web Cache Message Fields

Fields	Description
TIMESTAMP	The date and time when the message was generated. Time is either displayed in local or Greenwich Mean Time.
MSG_TYPE	The type of message. Possible values are NOTIFICATION, WARNING, TRACE, and DEBUG.
MSG_ID	The ID that uniquely identifies the message within the component. The ID consists of a 5-digit number. For example: 08513.
ECID	The Execution Context ID (ECID), which is a global unique identifier of the execution of a particular request in which the originating component participates. You can use the ECID to correlate error messages from different components. **See Also:** Section 9.1.1.4 for more information about the Oracle-ECID request header
MSG_TEXT	The text of the error message.

For example:

```
[04/Nov/2008:06:11:53 -0500] [notification 08513] Cache server process ID 13466is
starting up.
[04/Nov/2008:06:11:53 -0500] [notification 09612] [ecid: -] Oracle Web Cache 11g
(11.1.1)
[04/Nov/2008:06:11:53 -0500] [notification 13002] [ecid: -] Maximum allowed
incoming connections are 700
[04/Nov/2008:06:11:53 -0500] [notification 09446] [ecid: -] Statistics
initialization commencing.
[04/Nov/2008:06:11:53 -0500] [notification 09438] [ecid: -] The statistics
persistent repository is being reset by new configuration
[04/Nov/2008:06:11:53 -0500] [notification 09441] [ecid: -] DMS enabled
```

```
[04/Nov/2008:06:11:53 -0500] [notification 09447] [ecid: -] Statistics
initialization complete.
[04/Nov/2008:06:11:54 -0500] [notification 12209] [ecid: -] A 1 node cluster
successfully initialized
[04/Nov/2008:06:11:54 -0500] [notification 09614] [ecid: -] The following Oracle
Web Cache internal files are pre-populated to the cache: [[/host:port/_oracle_
http_server_webcache_static_.html]]
```

9.1.1.3 Request Details in Message 9720

Oracle Web Cache displays the request detail format in message 09720 when you enable option **Include Request Details** in the event log messages. This message is logged the first time an event is logged for a request with the following additional request details, including the client IP address, site name of the request and URL of the request.

Table 9–4 describes the fields for the request detail format.

Table 9–4 Request Details

Fields	Description
`[detail]`	Request detail event
`[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

For example:

```
[2008-11-20T23:27:32Z] [webcache] [TRACE:1] [WXE-09720] [io] [ecid: 15431471130,0]
[req-info: ] [client: 140.87.8.166] [host: -] [url: /images/image1k.bmp]
[2008-11-20T23:27:31Z] [webcache] [TRACE:1] [WXE-11331] [frontend] [ecid:
15431471130,0] Request matches configured site: www.company.com:80
[2008-11-20T23:27:31Z] [webcache] [TRACE:1] [WXE-11414] [population] [ecid:
15431471130,0] Basic cache key is composed with sitename www.company.com:80, URI
/images/image1k.bmp, method GET, post body -.
[2008-11-20T23:27:31Z] [webcache] [TRACE:1] [WXE-11304] [frontend] [ecid:
15431471130,0] Cache miss 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.

9.1.1.4 About the Oracle-ECID Request-Header Field

The `Oracle-ECID` request header is used to track requests as they move through the Oracle Fusion Middleware architecture. This information is especially useful for diagnostic purposes. Because Oracle 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 Fusion Middleware. Oracle Web Cache typically assigns an initial sequence number of 0 to a request. As a request passes from Oracle Web Cache to other Oracle Fusion Middleware components, the request ID remains constant, but the sequence number increments with each hop.

You can configure Oracle 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 event logs, you enable the **Include Request Details** option, 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 Fusion Middleware stops for the same request.

Oracle 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 Section 8.8 or further information about configuring diagnostic output in the `Server` response-header field or the HTTP response message that includes `Oracle-ECID` information

9.1.2 Event Log Examples

This section contains the following event log examples:

- Section 9.1.2.1, "Example: Event Log with Unsuccessful Startup Entries"
- Section 9.1.2.2, "Example: Event Log with Shutdown Entries"
- Section 9.1.2.3, "Example: Event Log with Cache Miss and Cache Hit Entries"
- Section 9.1.2.4, "Example: Event Log with an Invalidation Entry"
- Section 9.1.2.5, "Example: Analyzing ESI Events"

9.1.2.1 Example: Event Log with Unsuccessful Startup Entries

The following shows an event log excerpt with unsuccessful startup events. Oracle Web Cache cannot listen on port 7777, because it is in use. These errors can occur if Oracle Web Cache is running and listening on that port or another application is using that port.

```
[2008-11-04T16:37:24-05:00] [webcache] [NOTIFICATION:1] [WXE-08513] [logging]
[ecid: ] Cache server process ID 2427 is starting up.
[2008-11-04T16:37:24-05:00] [webcache] [NOTIFICATION:1] [WXE-09612] [main]
[ecid: ] Oracle Web Cache 11g (11.1.1)
[2008-11-04T16:37:24-05:00] [webcache] [NOTIFICATION:1] [WXE-13002] [config]
[ecid: ] Maximum allowed incoming connections are 700
[2008-11-04T16:37:24-05:00] [webcache] [NOTIFICATION:1] [WXE-09446] [stats]
[ecid: ] Statistics initialization commencing.
[2008-11-04T16:37:24-05:00] [webcache] [NOTIFICATION:1] [WXE-09438] [stats]
[ecid: ] The statistics persistent repository is being reset by new configuration
[2008-11-04T16:37:24-05:00] [webcache] [NOTIFICATION:1] [WXE-09441] [stats]
[ecid: ] DMS enabled
[2008-11-04T16:37:24-05:00] [webcache] [NOTIFICATION:1] [WXE-09447] [stats]
[ecid: ] Statistics initialization complete.
[2008-11-04T16:37:25-05:00] [webcache] [TRACE:1] [WXE-11366] [frontend] [ecid: ] A
client connection to listening port 7777 is dropped.
[2008-11-04T16:37:25-05:00] [webcache] [TRACE:1] [WXE-11380] [frontend] [ecid: ]
Network failure during client listen client listen (details: internal=failure
system=2)
[2008-11-04T16:37:25-05:00] [webcache] [ERROR:1] [WXE-09707] [main] [ecid: ]
Failed to start the server.
[2008-11-04T16:37:25-05:00] [webcache] [ERROR:1] [WXE-09609] [main] [ecid: ] The
server process could not initialize.
```

```
[2008-11-04T16:37:25-05:00] [webcache] [NOTIFICATION:1] [WXE-09610] [main]
[ecid: ] The server is exiting.
[2008-11-04T16:37:25-05:00] [webcache] [NOTIFICATION:1] [WXE-08514] [logging]
[ecid: ] Cache server process ID 2427 is shutting down.
```

9.1.2.2 Example: Event Log with Shutdown Entries

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

```
[2008-11-04T16:19:58-05:00] [webcache] [NOTIFICATION:1] [WXE-09703] [main]
[ecid: ] Stop Issued. The program will shut down after all accepted requests are
served, or a timeout occurs.
[2008-11-04T16:21:29-05:00] [webcache] [NOTIFICATION:1] [WXE-09610] [main]
[ecid: ] The server is exiting.
```

9.1.2.3 Example: Event Log with Cache Miss and Cache Hit Entries

The following shows an event log excerpt containing events for a cache-miss request:

```
[2008-11-04T15:37:02-05:00] [webcache] [TRACE:1] [WXE-11331] [frontend] [ecid:
5415484202,0] Request matches configured site: www.company.com:80
[2008-11-04T15:37:02-05:00] [webcache] [TRACE:1] [WXE-11414] [population] [ecid:
5415484202,0] Basic cache key is composed with sitename www.company.com:80, URI
/invalidate1/tcal_fct_invalidate_basic_2.html, method GET, post body -.
[2008-11-04T15:37:02-05:00] [webcache] [TRACE:1] [WXE-11304] [frontend] [ecid:
5415484202,0] Cache miss request.
[2008-11-04T15:37:02-05:00] [webcache] [TRACE:1] [WXE-11224] [os] [ecid:
5415484202,0] Site localhost:8888 matches site-to-server mapping
www.company.com:80.
[2008-11-04T15:37:02-05:00] [webcache] [TRACE:1] [WXE-11227] [os] [ecid:
5415484202,0] Initial Request is routed to origin server host-server:8080 using
load balancing.
[2008-11-04T15:37:02-05:00] [webcache] [TRACE:1] [WXE-11403] [population] [ecid:
5415484202,0]  begin cacheability decision for url:
www.company.com:80/invalidate1/tcal_fct_invalidate_basic_2.html
[2008-11-04T15:37:02-05:00] [webcache] [TRACE:1] [WXE-11481] [population] [ecid:
5415484202,0] Request/Response matches caching rule with URL expression
"^/invalidate1/.*\.h.*$".
[2008-11-04T15:37:02-05:00] [webcache] [TRACE:1] [WXE-13736] [compression] [ecid:
5415484202,0] Compression is disabled because the browser does not support
compression.
[2008-11-04T15:37:02-05:00] [webcache] [TRACE:1] [WXE-11446] [population] [ecid:
5415484202,0] URL which will be cached is: www.company.com:80/invalidate1/tcal_
fct_invalidate_basic_2.html
[2008-11-04T15:37:02-05:00] [webcache] [TRACE:1] [WXE-11415] [population] [ecid:
5415484202,0] Final cache key is composed sitename www.company.com:80, URI
/invalidate1/tcal_fct_invalidate_basic_2.html, method GET, post body -,
multiversion -, compressed no.
[2008-11-04T15:37:02-05:00] [webcache] [TRACE:1] [WXE-11088] [backend] [ecid:
5415484202,0] Following URL is now in cache: www.company.com:80/invalidate1/tcal_
fct_invalidate_basic_2.html
```

The following shows an event log excerpt containing events for a subsequent cache-hit request:

```
[2008-11-04T15:37:39-05:00] [webcache] [TRACE:1] [WXE-09720] [frontend] [ecid:
417732382502,0] [req-info: ] [client: 127.0.0.1] [host: www.company.com:80] [url:
/x-oracle-cache-invalidate]
[2008-11-04T15:37:39-05:00] [webcache] [TRACE:1] [WXE-11331] [frontend] [ecid:
417732382502,0] Request matches configured site: localhost:8888
[2008-11-04T15:37:39-05:00] [webcache] [TRACE:1] [WXE-11414] [population] [ecid:
417732382502,0] Basic cache key is composed with sitename www.company.com:80, URI
```

```
/invalidate1/tcal_fct_invalidate_basic_5.html, method GET, post body -.
[2008-11-04T15:37:39-05:00] [webcache] [NOTIFICATION:1] [WXE-11707] [invalidation]
[ecid: 417732382502,0] Object with URL '/invalidate1/tcal_fct_invalidate_basic_
5.html' is successfully invalidated.
[2008-11-04T15:37:39-05:00] [webcache] [NOTIFICATION:1] [WXE-11748] [invalidation]
[ecid: 417732382502,0] Invalidation with INFO 'about-ttl' has returned with status
'SUCCESS'; number of objects invalidated: '1'.
```

9.1.2.4 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 `/invalidation1/tcal_fct_invalidate_basic_5.html`.

```
[2008-11-04T15:37:39-05:00] [webcache] [TRACE:1] [WXE-09720] [frontend] [ecid:
417732382502,0] [req-info: ] [client: 10.10.150.35] [host: host:port] [url:
/x-oracle-cache-invalidate]
[2008-11-04T15:37:39-05:00] [webcache] [TRACE:1] [WXE-11331] [frontend] [ecid:
417732382502,0] Request matches configured site: www.company.com:80
[2008-11-04T15:37:39-05:00] [webcache] [TRACE:1] [WXE-11414] [population] [ecid:
417732382502,0] Basic cache key is composed with sitename localhost:8888, URI
/invalidate1/tcal_fct_invalidate_basic_5.html, method GET, post body -.
[2008-11-04T15:37:39-05:00] [webcache] [NOTIFICATION:1] [WXE-11707] [invalidation]
[ecid: 417732382502,0] Object with URL '/invalidate1/tcal_fct_invalidate_basic_
5.html' is successfully invalidated.
[2008-11-04T15:37:39-05:00] [webcache] [NOTIFICATION:1] [WXE-11748] [invalidation]
[ecid: 417732382502,0] Invalidation with INFO 'about-ttl' has returned with status
'SUCCESS'; number of objects invalidated: '1'.
```

9.1.2.5 Example: Analyzing ESI Events

The following provides an example of the messages in the event log for an ESI fragment for a cache miss. The messages in the event log report information about:

- How Oracle Web Cache processes ESI in the template
- How ESI processing loads an ESI fragment
- After the fragment is loaded, how the caching decision for an ESI fragment is formed. It includes information regarding the reason the fragment is cached or not cached.

In the following examples, `TRACE:1` messages are for the `verbosity=TRACE` level and `TRACE:32` messages are for the `verbosity=DEBUG` level. Setting verbosity to `DEBUG` includes `TRACE`, `NOTIFICATION`, `WARNING`, and `ERROR` level messages. `TRACE` includes `NOTIFICATION`, `WARNING`, and `ERROR`, but not `DEBUG`.

You do not see the following log messages shown in the following example unless you have the set the `event_log` verbosity level to `DEBUG`:

```
[2008-11-04T16:29:14-05:00] [webcache] [TRACE:1] [WXE-11952] [esi] [ecid:
211577120190,0] Start processing ESI document
www.company.com:80/cgi-bin/esi-headers.sh?/esi/esi-headers.html&localhost:8888,
nesting level 1 [2008-11-04T16:29:14-05:00] [webcache] [TRACE:1] [WXE-11331]
[frontend] [ecid: 211577120190,0] Request matches configured site:
www.company.com:80
[2008-11-04T16:29:14-05:00] [webcache] [TRACE:1] [WXE-11953] [esi] [ecid:
211577120190,0] In ESI template
www.company.com:80/cgi-bin/esi-headers.sh?/esi/esi-headers.html&localhost:8888,
the fragment's site name and URL has been discovered as www.company.com:80 and
/esi/include0.html [2008-11-04T16:29:14-05:00] [webcache] [TRACE:1] [WXE-11304]
[frontend] [ecid: 211577120190,0] Cache miss request.
```

```
[2008-11-04T16:29:14-05:00] [webcache] [TRACE:1] [WXE-11224] [os] [ecid:
211577120190,0] Site www.company.com:80 matches site-to-server mapping
www.company.com:80.
[2008-11-04T16:29:14-05:00] [webcache] [TRACE:1] [WXE-11227] [os] [ecid:
211577120190,0] Initial Request is routed to origin server
stadk61.us.oracle.com:8080 using load balancing.
[2008-11-04T16:29:14-05:00] [webcache] [TRACE:1] [WXE-11403] [population] [ecid:
211577120190,0]  [[ begin cacheability decision for
url: www.company.com:80/esi/include0.html ]]
[2008-11-04T16:29:14-05:00] [webcache] [TRACE:1] [WXE-11481] [population] [ecid:
211577120190,0] Request/Response matches caching rule with URL expression "/*".
[2008-11-04T16:29:14-05:00] [webcache] [TRACE:1] [WXE-11446] [population] [ecid:
211577120190,0]  [[ URL which will be cached is:
www.company.com:80/esi/include0.html ]]
[2008-11-04T16:29:14-05:00] [webcache] [TRACE:1] [WXE-11415] [population] [ecid:
211577120190,0] Final cache key is composed sitename www.company.com:80, URI
/esi/include0.html, method GET, post body -, multiversion -, compressed no.
[2008-11-04T16:29:14-05:00] [webcache] [TRACE:1] [WXE-11088] [backend] [ecid:
211577120190,0]  [[ Following URL is now in cache:
www.company.com:80/esi/include0.html ]]
```

9.2 Introduction to Access Logs

Oracle Web Cache records information about the received HTTP and HTTPS requests in access logs. Each Web site Web site defined in Oracle Web Cache can have its own access log. By default, the access log has a file name of `access_log` and is stored in the following directories:

(UNIX) *ORACLE_INSTANCE*/diagnostics/logs/WebCache/<*webcache_name*>
(Windows) *ORACLE_INSTANCE*\diagnostics\logs\WebCache\<*webcache_name*>

This section includes the following topics:

- Section 9.2.1, "Access Log Formats"
- Section 9.2.2, "Access Log Fields"
- Section 9.2.3, "Access Log Examples"

9.2.1 Access Log Formats

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, Oracle Web Cache provides support for the following access log formats:

- Section 9.2.1.1, "Common Log Format (CLF)"
- Section 9.2.1.2, "Enhanced CLF (ECLF)"
- Section 9.2.1.3, "Combined Log Format"
- Section 9.2.1.4, "Enhanced Combined Log Format"
- Section 9.2.1.5, "End-User Performance Monitoring Format"

9.2.1.1 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

9.2.1.2 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

9.2.1.3 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 must determine what kind of browser is sending the request, and where the browser was visiting before the request was forwarded to Oracle Web Cache.

9.2.1.4 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

9.2.1.5 End-User Performance Monitoring Format

This format provides support for the following fields intended for end-user performance monitoring of 10*g* 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)

9.2.2 Access Log Fields

If the default formats are not suitable for your environment, you can create custom log formats by specifying the fields that you require. Table 9–5 describes the supported fields. Fields prefixed with x or r are proprietary to Oracle Web Cache.

Table 9–5 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 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 9-20
cs-bytes	Bytes received from the client
cs-method	Client-to-Oracle Web Cache HTTP request method
cs-uri	Client-to-Oracle Web Cache URI
cs-uri-query	Client-to-Oracle Web Cache query portion of URI, omitting the stem
cs-uri_stem	Client-to-Oracle Web Cache stem portion of URI, omitting the query
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 Oracle 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 Oracle 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 Oracle Web Cache computer
sc(*header_name*)	HTTP response header sent from Oracle Web Cache to the client **See Also:** "cs(header_name) and sc(header_name) Access Log Fields" on page 9-20

Table 9–5 (Cont.) Access Log Fields

Field	Description
`sc-status`	Oracle 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 Oracle 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 Oracle 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 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 9–5 (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* is: ■ 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* is: ■ 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 Oracle 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 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 9–5 (Cont.) Access Log Fields

Field	Description
x-conn-abrt	Single character that specifies the whether 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 Oracle Web Cache could complete a response. ■ O specifies that the connection was terminated by the origin server before it could complete a response to Oracle Web Cache. ■ N specifies the response was completed without the connection being terminated.
x-date-start	Date before Oracle Web Cache received the first byte of the request, in the following format: *yyyy-mm-dd*
x-date-end	Date when Oracle 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:** Section 9.1.1.4 for further information about the Oracle-ECID request header
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 Oracle 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 Oracle 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 Oracle Web Cache response. ■ N specifies that Oracle 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. Oracle Web Cache cannot obtain the value for this field. Therefore, Oracle Web Cache displays a hyphen (-) in the output when this field is set.
x-os-name	Origin server or cache cluster member that Oracle 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 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 9–5 (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 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 Oracle 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 Oracle 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 Oracle Web Cache received the first byte of the request ■ The time immediately before Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 Oracle 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 9–5 (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 been sent to the origin server by Oracle Web Cache to update an existing object, Oracle 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 Oracle Web Cache back-end 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 Oracle Web Cache received the first byte of the request, in the following format: *hh:mm:ss:ssssss*

9.2.2.1 cs(*header_name*) and sc(*header_name*) Access Log Fields

Table 9–6 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 possible headers. It is not an exhaustive list.

Table 9–6 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 9–7 lists examples of cookie-related headers that can be used for the cs(*header_name*) and sc(*header_name*) fields.

Table 9–7 Supported Cookie-Related Header Fields

cs(*header_name*) Field	sc(*header_name*) Field
Cookie	Set-Cookie

Table 9–8 lists examples of Oracle Web Cache headers that can be used for the cs(*header_name*) and sc(*header_name*) fields.

Table 9–8 Supported Oracle Web Cache Header Fields

cs(*header_name*) Field	sc(*header_name*) Field
Surrogate-Capability	Surrogate-Control

9.2.3 Access Log Examples

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)

In addition, this section contains the following access log examples:

- Section 9.2.3.1, "Example: Access Log with Reload Entries"
- Section 9.2.3.2, "Example: Access Log with Status Code 404 Entry"
- Section 9.2.3.3, "Example: Access Log in Combined Format"
- Section 9.2.3.4, "Example: Access Log with Site Information"
- Section 9.2.3.5, "Example: Access Log with ESI Diagnostic Information"
- Section 9.2.3.6, "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
```

9.2.3.1 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.

9.2.3.2 Example: Access Log with Status Code 404 Entry

The following shows an access log excerpt in which Oracle 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
```

9.2.3.3 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)"
```

9.2.3.4 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 Oracle 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
```

9.2.3.5 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
```

9.2.3.6 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."
```

9.3 Configuring Event Logs

To configure event log settings:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Event Logs**.

   The Event Log Configuration page displays.
3. Specify the following settings for each cache in the **Cache-Specific Settings** table:
   a. In the **Directory** field, enter the directory in which to write event logs.

      By default, the event log is stored in the following directories:

      (UNIX) *ORACLE_INSTANCE*/diagnostics/logs/WebCache/<*webcache_name*>
      (Windows) *ORACLE_INSTANCE*\diagnostics\logs\WebCache\<*webcache_name*>

   b. Click **Enable Buffering?** to enable buffered logging; deselect the check box to disable buffered logging.

      With buffered logging, Oracle Web Cache stores log messages in memory. Oracle Web Cache writes them out in bulk to the event log when the buffer size or the flush interval is reached. Buffered logging increases performance by reducing the number of disk I/O operations.

      If the Oracle Web Cache server shuts down unexpectedly, buffered log messages may be lost.

      Oracle recommends disabling buffering to see the event log results immediately.

   c. If buffering is enabled, in the **Flush Interval** field, enter the interval, in seconds, when Oracle Web Cache writes contents of the buffer to the event log file.

The default is 10 seconds. When the interval is reached, Oracle Web Cache writes buffered information to the event log file. Even if the buffer is not full, Oracle Web Cache updates the event log. Oracle recommends not changing the default, unless you want to lower the interval to see results more frequently.

A value of 0 specifies that Oracle Web Cache will only flush the buffered event log when the specified buffer size has been exceeded.

d. If buffering is enabled, in the **Buffer Size** field, enter the size of the buffer, expressed in characters.

The default is 2048 characters. You can specify a maximum value of 32,768 characters.

e. From the **Verbosity** list, select the needed level of detail for the event log. The levels are described in Table 9–9.

Table 9–9 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 configuration. ▪ Site resolution ▪ Site-to-server mappings route to the correct origin servers ▪ Compression ▪ Session binding ▪ Caching rules ▪ ESI processing
Debug	Provides detailed events for troubleshooting. This level is intended for Oracle Support Services.

4. Set the global event log settings in the **Global Event Log Configuration** section:

a. In the **File Name** field, enter a name for the event log file.

The default file name is `event_log`.

b. From the **File Format** list, select the log format.

See Section 9.1.1 for further information about the formats.

c. From the **Time Style** list, select either **Local** or **GMT** (Greenwich Mean Time) to modify the format of 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 operating system. As such, Oracle Web Cache has no mechanism to improve the performance of the conversion process.

---

d. Click **Request-Based Logging** to enable request-based logging.

With request-based logging, Oracle Web Cache collects all event log messages associated with each request in memory. If the most severe message in the request is at or above the selected verbosity level, Oracle Web Cache writes all the messages related to the request to the event log at once. Oracle Web Cache groups the messages for the request together in the log file for easier diagnosis. For example, if verbosity is set to Notification, and Oracle Web Cache encounters an error at the Trace or Debug level, Oracle Web Cache writes all of the event log messages for the request to the event log.

Select **Disabled** to view results as they happen, especially when the verbosity is set to a level higher than Notification.

e. In **Include Request Details**, select **Yes** to enable Oracle Web Cache to write the information from the `Oracle-ECID` request header, or select **No** to not write request information to the event log. See Section 9.1.1.2 for further information about how request details are logged.

Select **No** if either of the following conditions apply:

- You are concerned about the performance impact of event log entries for request details.
- Oracle Web Cache is running in a standalone environment without Oracle HTTP Server.

5. Specify a rollover policy:

a. In the Rollover By Time section, click **Edit.**

The Edit Rollover Policy dialog displays.

You can use the rollover options in combination. For example, you can use both **Rollover by Time** and **Rollover by Size** or both **Retention by Size** and **Retention by Time**. Oracle Web Cache performs rollover based on whichever is reached first.

b. From the **Rollover by Time** list, select **Never**, **Hourly**, **Daily**, or **Weekly** to specify how often you want Oracle Web Cache to save current log information to *`event_log_file.yyyymmddhhmm`* and write future log information to a new log file with the configured log file name.

If you have a high-volume site, select **Daily** or **Hourly**.

c. In the **Scheduled Time** field, for **Hourly**, **Daily**, and **Weekly**, enter a new time in the left-hand side fields and menus and add it to the schedule by clicking **Add**. Table 9–10 describes specific configuration instructions for **Hourly**, **Daily**, and **Weekly**.

Table 9–10 Configuring Rollover By Time

Policy	To configure:
Hourly	1. Enter a value in the field to reflect the minutes after the hour. The default 0 means at the start of the hour. 2. Click **Add**. 3. From the **Time Style** list, select either **Local** or **GMT (Greenwich Mean Time)**.
Daily	1. Enter a value in the hours and minutes fields. The default 0 means at the start of the day. 2. Click **Add**. 3. From the **Time Style** list, select either **Local** or **GMT**.

Table 9–10 (Cont.) Configuring Rollover By Time

Policy	To configure:
Weekly	1. Add a time by selecting a day of the week, and entering values in the hours and minutes fields. The default 0 means at the start of the week. 2. Click **Add**. 3. From the **Time Style** list, select either **Local** or **GMT (Greenwich Mean Time)**.

To remove a time from the schedule list, select the time, and then click **Remove**. The value moves to the left list, where you can modify it.

See Section 9.8 for instructions on immediately rolling over log files.

d. In the **Rollover by Size** field, enter the maximum size of the log file size at which rollover occurs. Specify 0 for unlimited size.

e. In the **Retention by Time** field, specify how long to keep log files before purging the oldest ones.

In the **Every** field, enter the quantity and from the list of **Hours**, **Days**, **Weeks**, **Months**, **Years**, select the duration. A quantity of 0 means unlimited time, which means Oracle Web Cache does not retain files based on time.

f. In the **Retention by Size** field, enter the total size of all log files before purging oldest ones. Specify 0 for unlimited size.

This value must be larger than the value you specify for the **Rollover Size** field.

If neither **Retention by Time** or **Retention by Size** is set, then log files can grow without limits. The log files could end up consuming all available space on the disk where this file is located.

g. Click **OK**.

6. Click **Apply** and restart Oracle Web Cache. See Section 2.13.

9.4 Configuring Access Logs

To configure access log settings:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Access Logs**.

   The Access Log Configuration page displays.
3. Specify the following settings for each cache in the **Cache-Specific Settings** table:

   a. In the **Directory** field, enter the directory in which to write access logs.

   By default, the event log is stored in the following directories:

   ```
   (UNIX) ORACLE_INSTANCE/diagnostics/logs/WebCache/<webcache_name>
   (Windows) ORACLE_INSTANCE\diagnostics\logs\WebCache\<webcache_name>
   ```

   b. Click **Enable Logging?** to enable logging; deselect to disable logging.

   c. Click **Enable Buffering?** to enable buffered logging; deselect the check box to disable buffered logging.

With buffered logging, Oracle Web Cache stores log messages in memory. Oracle Web Cache writes them out in bulk to the access log when the buffer size or the flush interval is reached. The buffer size is set to 2048 bytes. Buffered logging increases performance by reducing the number of disk I/O operations.

If the Oracle Web Cache server shuts down unexpectedly, buffered log messages may be lost.

Oracle recommends disabling buffering to view access log results immediately.

d. If buffering is enabled, in the **Flush Interval** field, enter the interval, in seconds, when Oracle Web Cache writes contents of the buffer to the access log file.

The default is 10 seconds. When the interval is reached, Oracle Web Cache writes buffered information to the access log file. Even if the buffer is not full, Oracle Web Cache updates the access log. Oracle recommends not changing the default, unless you want to lower the interval to see results more frequently.

A value of 0 specifies that Oracle Web Cache will only flush the buffered access log when the specified buffer size has been exceeded.

4. Specify the following settings for each site in the **Site-Specific Settings** table:

a. If you want to apply the settings from the **Default Settings** row to this site, click **Use Default for all Sites Settings**. Deselect this checkbox to provide site-specific overrides in the other fields.

b. In the **File Name** field, enter a name for the access log file.

The default file name is `access_log`.

c. Click **Enable Logging** to enable logging for the site; deselect to disable logging for the site.

Site-specific logging only takes effect if logging is enabled for the cache. If you enable this option, ensure that it is also selected for the cache in Step 3b.

d. Select **Log ESI Fragment Requests?** 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 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.**

e. From the **Format Style** list, select an access log format.

See Section 9.2.1 for a description of the default formats and Section 9.5 to create a customized style for your environment.

f. From the **Rollover Policy** list, select a rollover policy to specify how often you want to change the frequency at which Oracle Web Cache saves current log information to *`access_log_file.yyyymmddhhmm`* 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.

See Section 9.6 to modify an existing policy or create a new rollover policy.

5. Click **Apply** and restart Oracle Web Cache. See Section 2.13.

9.5 Creating a Customized Access Log Format

If the default formats described in Section 9.2.1 are not suitable for your environment, create a new log format:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Access Logs**.

   The Access Log Configuration page displays.
3. Click the **Log Formats** tab, and click **Create.**

   The Create Log Format dialog box displays.
4. In the **Format Name** field, enter a unique name for the format, keeping the following restrictions in mind:
   - The format name cannot contain any spaces or special characters other than underscore (_).
   - The name must be unique among other format names, rollover policy names, and session names.
5. From the **Separator** list, select the separator to use for separating access log fields.
6. 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-2008 00:00:00
   #Fields: c-ip x-auth-id x-clf-date cs(Host x-req-line sc-status bytes
   ```

   See `http://www.w3.org/TR/WD-logfile.html` for further information about XLF directives.
7. In the **XLF Fields** section, select an access log field name from the **Field Name** list.

   See Table 9–5 for a listing of the supported access logs fields
8. If you select cs(header_name or sc `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 Table 9–6, Table 9–7, and Table 9–8 for a description of the headers allowed for `cs(`*`header_name`*`)` and `sc(`*`header_name`*`)`
9. Click **Add.**
10. Perform Steps 7 and 9 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.
11. Click **OK** to apply changes and return to the Access Log Configuration page.
12. Click **Apply** in the Access Log Configuration page to apply this change.

9.6 Creating a Customized Access Log Rollover Policy

To modify an existing rollover policy or create a new rollover policy:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Administration** and then **Access Logs**.

   The Access Log Configuration page displays.
3. Click the **Rollover Policies** tab, and click **Create.**

   The Create Rollover Policy dialog box displays.

   You can use the rollover options in combination. For example, you can use both **Rollover by Time** and **Rollover by Size** or both **Retention by Size** and **Retention by Time**. Oracle Web Cache performs rollover based on whichever is reached first.
4. In the **Policy Name** field, enter a unique name for the rollover policy, keeping the following restrictions in mind:
   - The policy name cannot contain any spaces or special characters other than underscore (_).
   - The name must be unique among other policy names, log format style names, and session names.
5. In the **Rollover by Time** section, select **Never**, **Hourly**, **Daily**, **Weekly** to specify how often you want Oracle Web Cache to save current log information to *access_log_file.yyyymmddhhmm* and write future log information to a new log file with the configured log file name.

   If you have a high-volume site, select **Daily** or **Hourly**.
6. In the **Rollover by Time** section, select **Never**, **Hourly**, **Daily**, **Weekly** to specify how often you want Oracle Web Cache to save current log information to *access_log_file.yyyymmddhhmm* and write future log information to a new log file with the configured log file name.

   If you have a high-volume site, select **Daily** or **Hourly**.

   For **Hourly**, **Daily**, and **Weekly**, enter a new time in the left-hand side fields and menus and add it to the schedule by clicking **Add**. Table 9–11 describes specific configuration instructions for **Hourly**, **Daily**, and **Weekly**.

Table 9–11 Configuring Rollover By Time

Policy	To configure:
Hourly	1. Enter a value in the field to reflect the minutes after the hour. The default 0 means at the start of the hour. 2. Click **Add**. 3. From the **Time Style** list, select either **Local** or **GMT (Greenwich Mean Time)**.
Daily	1. Enter a value in the hours and minutes fields. The default 0 means at the start of the day. 2. Click **Add**. 3. From the **Time Style** list, select either **Local** or **GMT**.
Weekly	1. Add a time by selecting a day of the week, and entering values in the hours and minutes fields. The default 0 means at the start of the week. 2. Click **Add**. 3. From the **Time Style** list, select either **Local** or **GMT (Greenwich Mean Time)**.

To remove a time from the schedule list, select the time, and then click **Remove**. The value moves to the left list, where you can modify it.

See Section 9.8 for instructions on immediately rolling over log files.

7. In the **Rollover by Size** field, enter the maximum size of the log file size at which rollover occurs. Specify 0 for unlimited size.
8. In the **Retention by Time** field, specify how long to keep log files before purging the oldest ones.

   In the **Every** field, enter the quantity and from the list of **Hours**, **Days**, **Weeks**, **Months**, **Years**, select the duration. A quantity of 0 means unlimited time, which means Oracle Web Cache does not retain files based on time.
9. In the **Retention by Size** field, enter the total size of all log files before purging oldest ones. Specify 0 for unlimited size.

   This value must be larger than the value you specify for the **Rollover Size** field.

   If neither **Retention by Time** or **Retention by Size** is set, then log files can grow without limits. The log files could end up consuming all available space on the disk where this file is located.
10. Click **OK** to apply changes and return to the Access Log Configuration page.
11. Click **Apply** in the Access Log Configuration page to apply this change.

9.7 Viewing Event Logs and Access Logs

To view events logs, use either the Fusion Middleware Control or the WLST `listLogs` command. See the following documentation resources:

- *Oracle Fusion Middleware Administrator's Guide* for details on the various tools for viewing event logs
- *Oracle Fusion Middleware Error Messages Reference* describes the event log messages in further detail.

To view access logs, use any text editor.

9.8 Rolling Over Event and Access Logs

In addition to configuring event and access log rollover frequency, you can immediately roll over event and access logs. During the rollover process, Oracle Web Cache saves current information to lot file and writes future log information to a new log file with the configured log file name.

To immediately roll over log files:

1. Navigate to the Web Cache Home page in Fusion Middleware Control. See Section 2.6.2.
2. From the **Web Cache** menu, select **Operations** and then **On Demand Rollover**.

9.9 Using Audit Logs

Oracle Web Cache supports the Common Audit Framework for providing a uniform system for administering audits across Oracle Fusion Middleware components. The audit log files generated by Oracle Web Cache processes provide important information that can help you identify and diagnose potential security performance and configuration issues.

Oracle Web Cache records the following events in the audit log:

- Startup and shutdown events
- Inter-cache communication events, such as:
  - Authentication or challenge events
  - Subscriber cache insertion to subscriber list (success or failure)
  - Invalid address information from subscriber
  - Remote or subscriber cache authentication event
  - Addition or removal of cluster cache member
- Request authentication events, such as:
  - Login to Oracle Web Cache ports
  - Denied request due to access control settings or request-filtering rules
  - Identity denied to access cached objects
  - Invalidation in response containing wrong Web site information
  - Client certificate failed
  - SSL connection denied because no client certificate was provided
  - SSL connection denied because client certificate presented was on the CRL
- Configuration services, such as:
  - Dynamic configuration changes applied
  - SSL handshake failed with the origin server
  - Authentication with the proxy server failed

For more information, see *Oracle Fusion Middleware Security Guide* for further information about using audit logs.

10

Configuring Common Deployment Scenarios

This chapter describes how to configure common deployment scenarios using Oracle Web Cache. It includes the following topics:

- Section 10.1, "Using Oracle Web Cache in a Common Deployment"
- Section 10.2, "Using a Cache Hierarchy for a Global Intranet Application"
- Section 10.3, "Using Oracle Web Cache for High Availability without a Hardware Load Balancer"

10.1 Using Oracle Web Cache in a Common Deployment

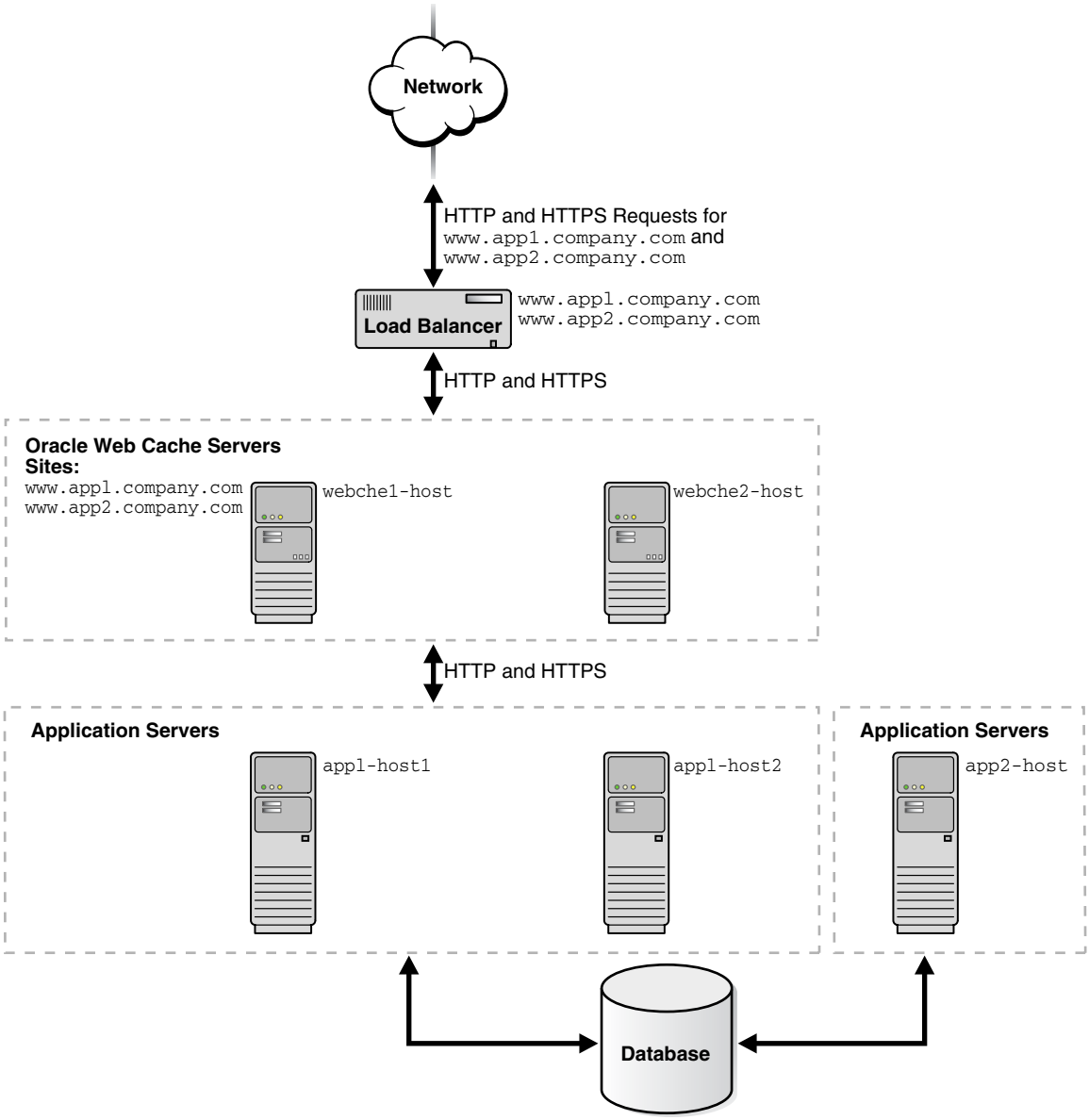
Figure 10–1 shows Oracle Web Cache in a common Oracle Application Server configuration. A tier of Oracle Web Cache servers cache content for a tier of application Web servers. The application Web servers `app1-host1` and `app1-host2` provide content for site `www.app1.company.com`, and `app2-host` provides content for `www.app2.company.com`. The two Oracle 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 Oracle Web Cache server on a periodic basis to check the status of the cache.

As a cache cluster, the two Oracle Web Cache servers provide failure detection and failover. If an Oracle 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. Oracle 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, Oracle Web Cache distributes the requests to an application Web server respective to the site.

As a failover pair, both Oracle Web Cache servers are configured to cache the same content. When both Oracle Web Cache servers are running, a load balancer distributes the load among both servers. If one server fails, the other server receives and processes all incoming requests.

Figure 10–1 Deploying Oracle 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 Oracle 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. Configure the load balancer with to ping each Oracle Web Cache server on a periodic basis with URL `/_oracle_http_server_webcache_static_.html`, which is stored in the cache.
4. If configuring a cache cluster, specify `webche1-host` and `webche2-host` as cluster members.

   See Section 3.6 for more information on configuring a cache cluster.

5. Configure the Oracle 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
   - Site definition for `www.app1.company.com` mapped to `app1-host1` and `app1-host2`
   - Site definition for `www.app2.company.com` mapped to `app2-host`

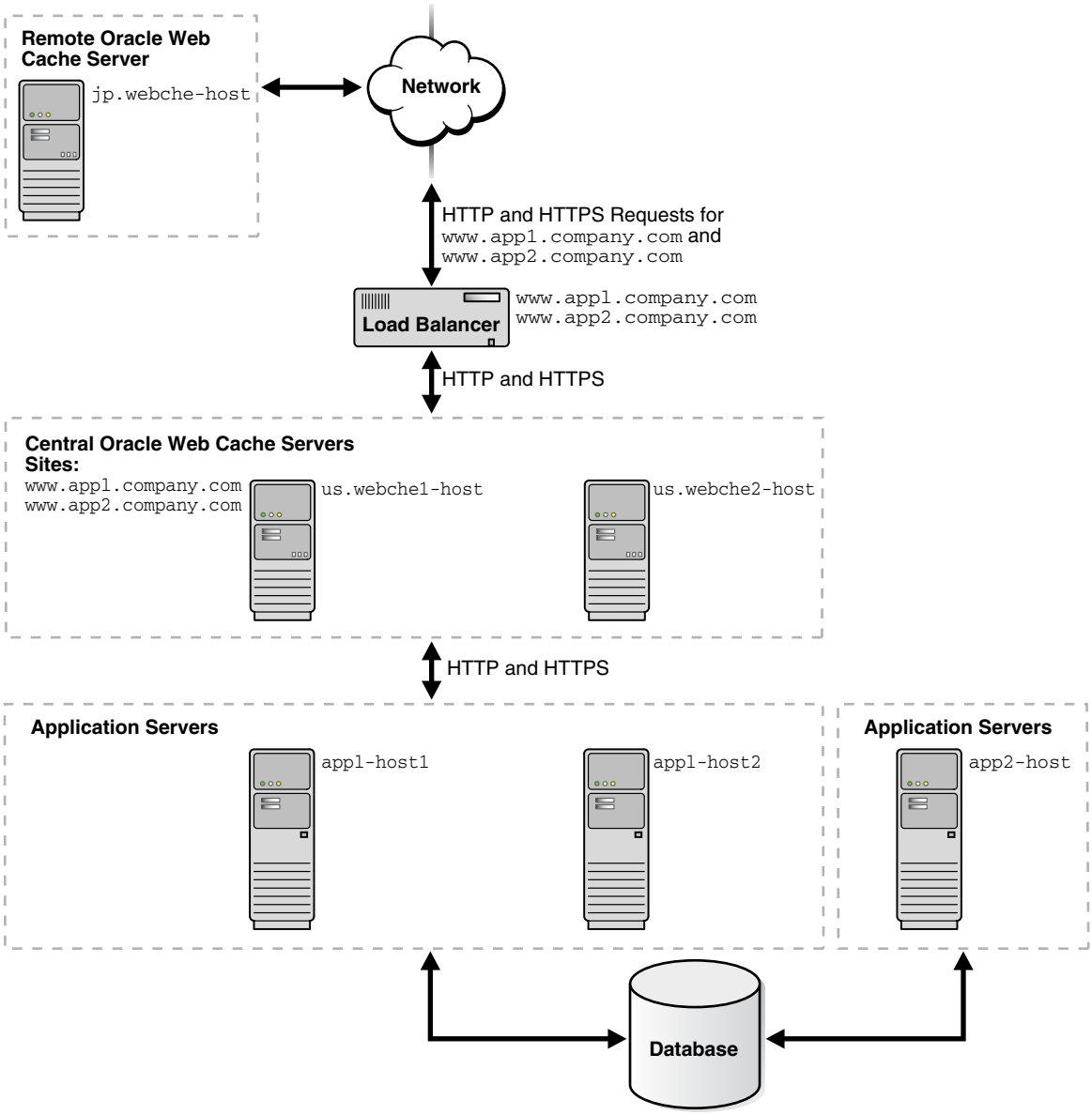
   For more information, see:
   - Section 2.11.1 for instructions about configuring listening ports
   - Section 2.11.2 for instructions about configuring origin server settings
   - Section 2.11.3 for instructions on creating site definitions and site-to-server mappings

10.2 Using a Cache Hierarchy for a Global Intranet Application

Many Web sites have several data centers. For networks with a distributed topology, you can deploy Oracle Web Cache at each of the data centers in a **distributed cache hierarchy**. Figure 10–2 on page 10-4 shows a distributed topology in which Oracle 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 Oracle Web Cache server to satisfy the request. Then, it returns the IP address of the appropriate Oracle Web Cache server to the client.

Figure 10–2 Deploying an Oracle 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 Oracle 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 Oracle 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

- Site definition for `www.app1.company.com` mapped to `app1-host1` and `app1-host2`
- Site definition for `www.app2.company.com` mapped to `app2-host`

4. Configure Oracle 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
   - Site definition for `www.app1.company.com` mapped to `app1-host1` and `app1-host2`
   - Site definition for `www.app2.company.com` mapped to `app2-host`
5. Enable propagation of invalidation messages for each of the caches in the cache hierarchy:
   1. Use a text editor to open `webcache.xml`, located in:

      (UNIX) *ORACLE_INSTANCE*/<*instance_name*>/config/WebCache/<*webcache_name*>
      (Windows) *ORACLE_INSTANCE*\<*instance_name*>\config\WebCache\<*webcache_name*>

   2. Locate the <INTERCACHE> element, a sub-element of the <SECURITY> element.
   3. Modify the ENABLEINBOUNDICC and ENABLEOUTBOUNDICC attributes to YES. For example:

```
<?xml version="1.0" encoding='ISO-8859-1'?>
<CALYPSO ... >
 <VERSION DTD_VERSION="11.1.1.0.0"/>
 <GENERAL>
   <CLUSTER NAME="WebCacheCluster" ... />
   <SECURITY SSLSESSIONTIMEOUT="3600" ... >
     <USER TYPE="INVALIDATION" ... />
     <USER TYPE="MONITORING" ... />
     <SECURESUBNET ALLOW="ALL"/>
     <DEBUGINFO HEADER="YES" ... />
     <HTTPREQUEST MAXTOTALHEADERSIZE="819000" ... />
     <INTERCACHE ENABLEINBOUNDICC="YES" ENABLEOUTBOUNDICC="YES"/>
   </SECURITY>
...
```

   4. Save `webcache.xml`.
6. Restart the caches in the hierarchy with the following command:

   opmnctl restartproc ias-component=*component_name*

   This executable is found in the following directory:

   (UNIX) *ORACLE_INSTANCE*/bin
   (Windows) *ORACLE_INSTANCE*\bin

For more information, see:

- Section 2.11.1 for instructions about configuring listening ports
- Section 2.11.2 for instructions about configuring origin server settings
- Section 2.11.3 for instructions on creating site definitions and site-to-server mappings

- Section 7.10.2.2 to understand how invalidation in a hierarchy works

10.3 Using Oracle Web Cache for High Availability without a Hardware Load Balancer

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

- Oracle 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 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).

For more information, see Section 3.8 and Section 3.9 for configuration details.

Part III

Advanced Administration

This part presents information about performing advanced administration tasks for Oracle Web Cache. It contains the following chapters:

- Chapter 11, "Caching Dynamic Content with ESI Language Tags"
- Chapter 12, "Caching with Third-Party Application Servers"

11

Caching Dynamic Content with ESI Language Tags

This chapter describes the **Edge Side Includes (ESI)** tags provided for content assembly of dynamic fragments.

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 Oracle Web Cache. A template page is configured with ESI markup tags that fetch and include dynamic HTML fragments. The fragments themselves can also contain ESI markup. You can assign caching rules to the template page and HTML fragments. By enabling page assembly in Oracle Web Cache rather than in the origin server, you can increase cache hit rates and improve performance.

This chapter includes the following topics:

See `http://www.esi.org` for the ESI language release 1.0 specification.

11.1 Introduction to ESI for Partial Page Caching

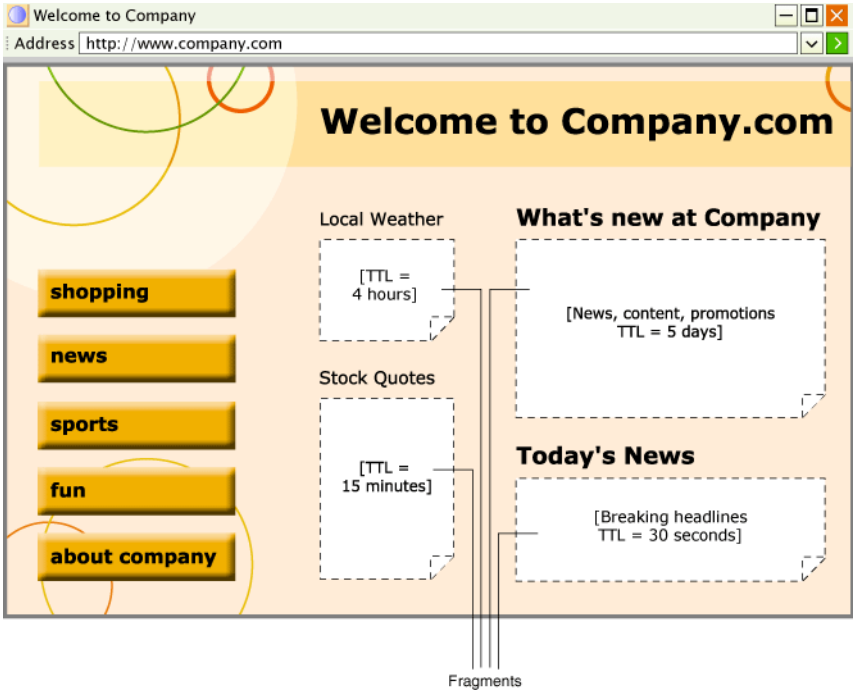
Oracle Web Cache provides dynamic assembly of Web pages with both cacheable and non-cacheable page fragments. It provides for assembly by enabling Web pages to be divided into fragments of differing caching profiles. These fragments are maintained as separate elements in the cache. The fragments are assembled into HTML pages as appropriate when requested by end users.

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

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 11–1, 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 11–1 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 Oracle Web Cache to fetch and include the HTML fragments. The fragments themselves are HTML files containing discrete text or other objects.

Each included fragment is a separate object with its own 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 11–1 provides a summary of the main ESI tags.

Table 11–1 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
`<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

Table 11–1 (Cont.) Summary of ESI Tags

Tag	Description
<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 11–1 shows the ESI markup language for the template page shown in Figure 11–1.

Example 11–1 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 11–2 shows the XML response of `GetProfile.jsp`, which provides access to profile environment variables.

Example 11–2 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>
```

11.1.1 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. Oracle Web Cache receives a request for a template page.
  2. Oracle 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.

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

- XML conversion to HTML

  Oracle Web Cache uses XSL Transformations (XSLT) to transform XML fragments into HTML.

11.1.1.1 ESI for Java (JESI)

Oracle Web Cache provides the JESI tag library as a convenient interface to ESI tags and functionality. In addition, you can deploy the JESI tag library on Oracle WebLogic Server. 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 if an ESI processor, such as Oracle 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.

For further information about using JESI with Oracle WebLogic Server, see:

http://www.oracle.com/technology/sample_code/tech/java/codesnippet/webcache/index.html

11.1.2 ESI Language Elements in the Surrogate-Control Response Header

Oracle Web Cache supports the ESI language tags, elements, and attributes listed in Table 11–2. 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 Oracle 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 have 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 Section 6.10 for further information about configuring the `Surrogate-Control` response header.

Table 11–2 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	Section 11.4.1	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI/1.0"`

`<esi:comment>` tag	Section 11.4.2	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI/1.0"`

`<esi:environment>` tag	Section 11.4.3	`"ORAESI/9.0.4"` `"ORAESI/9.0.2"`

`<esi:include>` tag	Section 11.4.4	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI/1.0"`

`<esi:environment>` tag and `<esi:include>` tag attributes and elements		
`alt` attribute	Section 11.4.4	`"ORAESI/9.0.4"` `"ESI/1.0"`
`max-age` attribute	Section 11.4.4 Section 11.4.3	`"ORAESI/9.0.4"` `"ORAESI/9.0.2"`

`onerror` attribute	Section 11.4.4	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI/1.0"`

Table 11–2 (Cont.) ESI Language Features

ESI Language Feature	See Also	content="*value*" Control Directive in Surrogate-Control Response Header Supporting Feature

`src` attribute	Section 11.4.4	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI/1.0"`

`timeout` attribute	Section 11.4.4 Section 11.4.3	`"ORAESI/9.0.4"` `"ORAESI/9.0.2"`
`<esi:log>` element	Section 11.4.4 Section 11.4.3	`"ORAESI/9.0.4"`
`<esi:request_header>` element	Section 11.4.4 Section 11.4.3	`"ORAESI/9.0.4"` `"ORAESI/9.0.2"`
`<esi:request_body>` element	Section 11.4.4 Section 11.4.3	`"ORAESI/9.0.4"` `"ORAESI/9.0.2"`

`<esi:inline>` tag	Section 11.4.5	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI-Inline/1.0"`
		`"ESI/1.0"`

`name` attribute	Section 11.4.5	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI-Inline/1.0"`
		`"ESI/1.0"`

`fetchable` attribute	Section 11.4.5	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI-Inline/1.0"`
		`"ESI/1.0"`

`max-age` attribute	Section 11.4.5	`"ORAESI/9.0.4"` `"ORAESI/9.0.2"`
`timeout` attribute	Section 11.4.4 Section 11.4.3	`"ORAESI/9.0.4"` `"ORAESI/9.0.2"`
`<esi:invalidate>` tag	Section 11.4.6	`"ESI-INV/1.0"`

`<esi:remove>` tag	Section 11.4.7	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI/1.0"`

`<esi:try>` | `<esi:attempt>` | `<esi:except>` tags	Section 11.4.8	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI/1.0"`

`<esi:vars>` tag	Section 11.4.9	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI/1.0"`

`<!--esi...--->` tag	Section 11.4.10	`"ORAESI/9.0.4"`
		`"ORAESI/9.0.2"`
		`"ESI/1.0"`

See `http://www.esi.org/spec.html` for the *ESI Language Specification 1.0* and the *Edge Architecture Specification*.

11.1.3 About the Surrogate-Control Response Header and Surrogate-Capability Request Header for Cached Objects

To enable Oracle 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.

```
Surrogate-Control: content="ESI-INV/1.0 ORAESI/9.0.4"
```

For each requested object from the cache, Oracle 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 Oracle Web Cache
- Identifies the types of ESI operations that Oracle Web Cache can perform

The `Surrogate-Capability` request-header enables Oracle Web Cache to identify the operations it can perform 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 Oracle 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 Oracle 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 Table 11–3 or further information about the ESI tags supported for each *`operation_value`*.

11.1.4 Syntax Rules

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

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

- ESI tags and attributes 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"/>
```

11.1.5 Nesting Elements

As shown in Example 11–3, an ESI tag can contain nested ESI elements and other HTML markup.

Example 11–3 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>
```

11.1.6 Variable Expressions

ESI supports the HTTP request variables and environment variables used with the <esi:environment> tag.

This section contains the following topics:

- Section 11.1.6.1, "Variable Usage"
- Section 11.1.6.2, "Variable Default Values"
- Section 11.1.6.3, "HTTP Request Variables"

See Section 11.4.3 for instructions on including custom variables

11.1.6.1 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>
```

11.1.6.2 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 Oracle Web Cache fetching `http://example.com/default.html` if the cookie `id` is not in the request:

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

As with other literals, if whitespace must 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.

11.1.6.3 HTTP Request Variables

Table 11–3 on page 11-11 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 11–3 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`

Table 11–3 (Cont.) HTTP Request Variables Supported by ESI

Variable Name	HTTP Header Field	Substructure Type/Variable Type	Description	Example

`$HTTP_HOST`	`Host` request-header field
	Specifies the host name and port number of the resource. Port 80 is the default port number.
		Not Applicable/ String	Returns the value of the `HOST` header	Variable Setting:
				`$(HTTP_HOST)`
				HTTP Request Header Contains:
				`Host:http://www.company.com:80`
				Result: Returns
				`http://www.company.com:80`

`$HTTP_REFERER`	`Referer` request-header field
	Specifies the URL of the reference resource
		Not Applicable/ String	Returns the value of the `REFERER` header	Variable Setting:
				`$(HTTP_REFERER)`
				HTTP Request Header Contains:
				`Referer: http://www.company.com:80`
				Result: Returns
				`http://www.company.com:80`

`$(HTTP_USER_AGENT{browser})`
`$HTTP_USER_AGENT{version})`
`$HTTP_USER_AGENT{os})`
	`User-Agent` request-header field
	Specifies the Web browser type, browser version, or operating system that initiated the request.
		Dictionary/ String	Specifies one of three keys: `browser` for browser type, `version` for browser version, and `os` for operating system	Variable Setting:
				`$(HTTP_USER_AGENT{browser})`
				HTTP Request Header Contains:
				`User-Agent:Mozilla/4.0 (compatible, MSIE 5.5, Windows)`
				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`.

Table 11–3 (Cont.) HTTP Request Variables Supported by ESI

Variable Name	HTTP Header Field	Substructure Type/Variable Type	Description	Example

`$(QUERY_STRING)`	Not Applicable	Not Applicable/ String	Specifies to return the entire query string encoded	Variable Setting:
				`$(QUERY_STRING)`
				Query Request Contains:
				`CEO=Jane%20Doe&CFO=John %20Doe`
				Result: Returns the entire query string encoded:
				`CEO=Jane%20Doe&CFO=John %20Doe`

`$(QUERY_STRING_ ENCODED{`*parameter*`})`	Not Applicable	Dictionary/ String	Given a parameter name in a query string, returns the value of the parameter with URL encoding. The query string can be in an URL or a request body.	Variable Setting:
				`$(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.`

11.1.7 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 for 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, Oracle Web Cache displays a default page for the fragment.

See the following sections:

- Section 11.4.4, "ESI include Tag"
- Section 11.4.8, "ESI try | attempt | except Tags"
- Section 2.11.6, "Task 6: Configure Error Pages"

11.1.8 About 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.

- Section 11.1.8.1, "Using Inline for Non-Fetchable Fragmentation"
- Section 11.1.8.2, "Using Inline for Fetchable Fragmentation"
- Section 11.1.8.3, "Using Include for Fragmentation"
- Section 11.1.8.4, "Selecting the Fragmentation Mechanism for Your Application"

11.1.8.1 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. Oracle 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 11–4 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 11–4 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.

11.1.8.2 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 11–5 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 11–5 Inline Fetchable Example

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

See Section 11.1.8.2 for further information about the fetchable attribute.

11.1.8.3 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 equals 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, Oracle 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 11–1 on page 11-3 shows ESI markup with `<esi:include>` tags.

11.1.8.4 Selecting the Fragmentation Mechanism for Your Application

Although both `<esi:include>` and `<esi:inline>` enable Oracle 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.

11.1.9 Referer Request-Header Field

When Oracle 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 Oracle 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.

11.1.10 Cookie Management for Template Pages and Fragments

Session cookie establishment for ESI templates and fragments works much the same way as typical Oracle 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 Oracle 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, Oracle 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, Oracle 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, Oracle Web Cache does not accumulate the `Set-Cookie` response-header field with that of the template and other matching fragments.

See Section 11.1.6 for a description of how you can use the `HTTP_COOKIE` variable in ESI markup.

11.2 Enabling Dynamic Assembly of Content and Partial Page Caching

For an overview of partial page caching, see Section 11.1.

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:

- Section 11.2.1, "Enabling Partial Page Caching"
- Section 11.2.2, "Using ESI for Simple Personalization"
- Section 11.2.3, "Examples of ESI Usage"

11.2.1 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 Section 11.2.2, "Using ESI for Simple Personalization".

   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.

For more information, see:

- Section 11.4 for further information about ESI markup tags
- Section 6.10 for further information about configuring the `Surrogate-Control` response-header field
- Section 6.8 for further information about configuring caching rules

11.2.2 Using ESI for Simple Personalization

You can use variable expressions for simple personalization.

For example, the following HTML substitutes 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.

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. You can also use variables with other ESI tags.

See the following sections:

- Section 11.1.6, "Variable Expressions"
- Section 11.4.9, "ESI vars Tag"

11.2.3 Examples of ESI Usage

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

- Section 11.2.3.1, "Example of a Portal Site Implementation"
- Section 11.2.3.2, "Example of Simple Personalization with Variable Expressions"

11.2.3.1 Example of a Portal Site Implementation

Figure 11–2 shows a portal site response page, `http://www.company.com/servlet/oportal?username=Mark`, for a registered user named Mark.

Figure 11–2 Portal Site Page

This page is assembled by Oracle 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, Oracle 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:

- Section 11.2.3.1.1, "Portal Example Using inline Tags"
- Section 11.2.3.1.2, "Portal Example Using Include Tags"

11.2.3.1.1 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 Section 11.4.5 for further information about the `fetchable` attribute.

To use 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 11–6 shows portal.esi with <esi:inline> tags.

Example 11–6 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 11–7 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 11–7 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 11–8 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 11–8 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 11–9 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 11–9 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 11–10 shows the markup for referencing an advertisement in the Promotion section. `promotionID` is the based on the user's identification.

Example 11–10 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 use ESI and dynamic content caching for these real-time fragments, use the `<esi:include>` tag.

See Section 11.2.3.1.2 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 11–11 shows the markup.

Example 11–11 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>
```

11.2.3.1.2 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 11–12 shows `portal.esi` with `<esi:include>` tags.

Example 11–12 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 11–13 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 11–13 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 11–14 shows `GetProfile`, which provides access to the `city`, `state`, `news`, and `sports` environment variables.

Example 11–14 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 11–15 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 11–15 portal.esi Example with vars tag: Personalized Greeting

```
<esi:vars>Welcome, $(QUERY_STRING{username})!</esi:vars>
```

Example 11–16 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 11–16 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 11–17. 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 11–17 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 11–18. It includes fragments for three stock quotes: IBM, ORCL, and YHOO.

Example 11–18 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 11–19. It includes fragments for three stock quotes: IBM, ORCL, and SCO. As 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 11–19 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 11–20 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, Oracle Web Cache can cache the template and integrate the dynamic advertisement into the template.

Example 11–20 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 11–21, the response to the included image fragment for the banner is not cacheable. When a user requests this page, Oracle 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 11–21 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 11–22, the next time the user reloads the page, `Advert` generates another banner for the request.

Example 11–22 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 11–23 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 11–23 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>
```

11.2.3.2 Example of Simple Personalization with Variable Expressions

ESI variables can be used within an HTML tag. For example, consider Example 11–24. Its HTML code uses PL/SQL for an HTML form with a text box in it.

Example 11–24 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 11–25 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 11–25 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>');
```

11.3 Using 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 multiple requests, but only the first one processes. The execution of the inline invalidation is blocking. That is, if the ESI template contains other ESI features, inline invalidation is 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>
```

For more information, see:

- Section 7.5.1 for invalidation request syntax
- Section 11.4.6 for a description of the `<esi:invalidate>` tag

11.3.1 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. When this page is requested, `show_bike.pl` is no longer valid—an invalidation is needed.

Example 11–26 shows the code for `show_bike.pl`.

Example 11–26 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 Oracle Web Cache to only cache this page for up to an hour.

Example 11–27 shows the code for `buy_bike.pl` with an inline invalidation request.

Example 11–27 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 Oracle Web Cache to process the `<esi:invalidate>` tag.

Example 11–28 shows the browser response for `buy_bike.pl`. Because Oracle Web Cache has already processed the inline invalidation request, the inline invalidation is not present in the response.

Example 11–28 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, Oracle Web Cache includes the invalidation request in the response.

- Enable the `output` attribute of the `<esi:invalidate>` tag.

  When the `output` attribute is enabled, Oracle Web Cache includes the invalidation result in the response enclosed within comments `<!--result-->`.

Example 11–29 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 11–29 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>
```

11.4 ESI Tag Descriptions

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

- Section 11.4.1, "ESI choose | when | otherwise Tags"
- Section 11.4.2, "ESI comment Tag"
- Section 11.4.3, "ESI environment Tag"
- Section 11.4.4, "ESI include Tag"
- Section 11.4.5, "ESI inline Tag"
- Section 11.4.6, "ESI invalidate Tag"
- Section 11.4.7, "ESI remove Tag"
- Section 11.4.8, "ESI try | attempt | except Tags"
- Section 11.4.9, "ESI vars Tag"
- Section 11.4.10, "ESI <!--esi-->Tag"

11.4.1 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.

11.4.1.1 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>
```

11.4.1.2 Attributes

`test`—Specifies the Boolean operation

11.4.1.3 Usage

- Each `<esi:choose>` tag must have a least one `<esi:when>` tag, and may optionally contain exactly one `<esi:otherwise>` tag.
- Oracle Web Cache executes 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 executes.
- Other HTML or ESI element can be included inside `<esi:when>` or `<esi:otherwise>` elements.

11.4.1.4 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 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")
```

11.4.1.5 Statements

Statements must be placed inside a `<esi:when>` or `<esi:otherwise>` subtag. Statements outside the subtags cannot be evaluated as conditions. Example 11–30 shows invalid placement of statements.

Example 11–30 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>
```

11.4.1.6 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>
```

11.4.2 ESI comment Tag

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

11.4.2.1 Syntax

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

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

11.4.2.2 Usage

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

11.4.2.3 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" />
```

11.4.3 ESI environment Tag

The `<esi:environment>` tag enables you to include custom environment variables from included fragments. When included, these variables can then be used with the other ESI tags.

11.4.3.1 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>
```

11.4.3.2 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 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.

11.4.3.3 Elements

- `request_body`—Specifies the HTTP request body of the fragment.
- `request_header`—Specifies an **HTTP request header** field and value for Oracle 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 Table 9–5 on page 9-15 for further information about the `x-esi-info` log field.

11.4.3.4 Syntax Usage

- Specify only one `<esi:environment>` tag for each template page, before other ESI tags.
- The attributes do not have to be in a particular order.
- Do not specify multiple `request_body` elements.
- 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, Oracle Web Cache uses other request headers from the parent page.
- Do not specify multiple `log` elements.

For more information, see:

- Section 11.1.6 for usage instructions for variables
- Section 11.4.4 for usage notes on `max-age`, `method`, `onerror`, `request_body`, and `request_header`
- Section 11.1.7 for usage notes on `onerror`

11.4.3.5 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>
```

11.4.4 ESI include Tag

The <esi:include> tag provides syntax for including fragments.

See Section 11.1.8 for a comparison of <esi:inline> and <esi:include> usage.

11.4.4.1 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>
```

11.4.4.2 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 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 if the XML file fragment is valid XML. For example, the following specifies that Oracle Web Cache use XSL Transformations (XSLT) to transform the XML into HTML using a style sheet. The style sheet 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 style sheet response pages are configured with a `Content-Type` response-header field that includes text and XML media types. For example:

```
Content-Type: text/xml
```

For more information about XSLT, see `http://www.w3.org/TR/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 for 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 Section 11.1.7 for usage notes on `alt` and `onerror`.

11.4.4.3 Elements

- `request_body`—Specifies the HTTP request body of the fragment.
- `request_header`—Specifies an HTTP request header field and value for Oracle 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 Table 9–5 for further information about the `x-esi-info` log field.

11.4.4.4 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 have to be in a particular order.
- The `src` attribute supports both HTTP and HTTPS. Oracle 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 Oracle Web Cache Manager, then Oracle Web Cache uses the protocol configured for the origin server in the Site-to-Server Mapping page. Oracle 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 multiple `request_body` elements.
- 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 multiple `log` elements.

11.4.4.5 Usage

The `<esi:include>` tag instructs Oracle 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.

Oracle 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 Oracle Web Cache.

If no request_header elements are specified, Oracle Web Cache uses other request headers from the parent page.

See Section 11.1.8 for a comparison of <esi:inline> and <esi:include> usage.

11.4.4.6 Examples

The following ESI markup includes a file named frag1.htm. The fragment must be fetched within 60 seconds. If the fetch fails, Oracle Web Cache ignores the includes and serves the page. If the fetch succeeds, Oracle Web Cache includes the fragment. Oracle 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" writes 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>
```

For more information, see Section 11.2.3.1 for an extended example of <esi:include> usage.

11.4.5 ESI inline Tag

The <esi:inline> tag marks a fragment as a separately cacheable fragment, embedded in the HTTP response of another object. Oracle Web Cache stores and assembles these fragments independently as <esi:include> fragments.

See Section 11.1.8 on page 11-14 for a comparison of <esi:inline> and <esi:include> usage.

11.4.5.1 Syntax

```
<esi:inline name="URL" fetchable="yes|no"
 [max-age="expiration_time [+ removal_time]"] [timeout="fetch_time"]
 Embedded HTML code
</esi:inline>
```

11.4.5.2 Attributes

- name—Specifies a unique name for the fragment in URL format.
- fetchable—yes instructs Oracle 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 Oracle 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.

11.4.5.3 Usage

Some inline fragments are only delivered as part of an HTTP response for another object. These are not independently fetchable by Oracle Web Cache the way <esi:include> fragments are. When a non-fetchable fragment is needed by Oracle Web Cache, it must request the object from which the inline fragment was extracted.

When a non-fetchable <esi:inline> fragment is not found in the cache, Oracle 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.

For more information, see:

- Section 11.1.8.1
- Section 11.1.8.2
- Section 11.4.4

11.4.5.4 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 Section 11.2.3.1 for an extended example of <esi:inline> usage.

11.4.6 ESI invalidate Tag

The <esi:invalidate> tag enables you to configure an invalidation request within the response of a browser page.

11.4.6.1 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>
```

11.4.6.2 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.

11.4.6.3 Usage

See Section 11.3.

11.4.6.4 Example

See Section 11.3.1.

11.4.7 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.

11.4.7.1 Syntax

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

11.4.7.2 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.

11.4.7.3 Example

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

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

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

11.4.8 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 Section 11.1.7 for usage notes on `alt` and `onerror`.

11.4.8.1 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>
```

11.4.8.2 Usage

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

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 Oracle Web Cache-to-origin server request queue limit being reached.
- noconnection: An error occurs because the cache cannot 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 Oracle 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 Oracle Web Cache.

11.4.8.3 Example

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

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

The following ESI markup attempts to fetch a fragment. If the fragment cannot be included because of `httpclienterror`, then Oracle 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, Oracle 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 Oracle 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 Oracle 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 Oracle 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>
```

11.4.9 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.

11.4.9.1 Syntax

<esi:vars>*Optional HTML code* $(*VARIABLE_NAME*{*key*}) *Optional HTML code*</esi:vars>

11.4.9.2 Syntax Usage

- If the variable does not use the complete `$(VARIABLE_NAME{key})` format, Oracle 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"/>
  ```

11.4.9.3 Usage

Section 11.1.6 and Section 11.4.3 for usage of HTTP request variables and custom variables

11.4.9.4 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>
```

11.4.10 ESI <!--esi-->Tag

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

11.4.10.1 Syntax

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

11.4.10.2 Usage

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

11.4.10.3 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.

12

Caching with Third-Party Application Servers

This chapter discusses how to configure Oracle Web Cache with third-party application Web servers.

This chapter includes the following topics:

Notes:

- While this chapter describes how Oracle Web Cache works with three specific kinds of servers, Oracle 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 requires more extensive configuration of Oracle Web Cache. Refer to the third-party application Web server documentation for information about designing applications.

12.1 Introduction to Third-Party Application Servers

Because Oracle Web Cache is transparent to the application Web server, the application Web server treats HTTP requests from Oracle 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 Oracle Web Cache as an HTTP message.

Because Oracle 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 Oracle 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.

12.1.1 Web Site Configuration

You configure Oracle 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 12–1 shows the default values for the listening ports for the products discussed in this appendix.

Table 12–1 Third-Party Application Web Server Default Listening Ports

Application Web Server	Port
IBM WebSphere Application Server, Version 6.0	80
Apache Tomcat, Version 4.1	8080
Microsoft IIS 6.0	80

To configure Oracle Web Cache to communicate with a third-party application Web server, perform the following tasks:

1. See Section 2.11.1 to change Oracle Web Cache port settings
2. See Section 2.11.2 configure application Web server settings
3. See Section 2.11.3 and Section 2.11.4 configure Web site settings

12.1.2 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 Chapter 6, "Caching and Compressing Content," and Chapter 7, "Invalidating Content."

12.2 IBM WebSphere

When Oracle 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 Oracle Web Cache `Surrogate-Capability: orcl="ESI/1.0"` request. Oracle Web Cache ignores the ESI/1.0+ features to get rid of page rendering issues and errors. If Oracle Web Cache is deployed as a caching solution, this difference in the control directive value may result in undefined Web application behavior.

Follow these steps to force IBM Websphere Application Server to send ESI/1.0 instead of ESI/1.0 or later:

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 an 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 several JSP, Java servlets, and EJB examples. This section explains how to configure Oracle Web Cache to cache the following content:

- Section 12.2.1, "WebSphere Snoop Servlet"
- Section 12.2.2, "WebSphere Calendar Creator JSP"

12.2.1 WebSphere Snoop Servlet

The `snoop` servlet shows getting and using request information, headers, and parameters sent by the browser. Use it to demonstrate how Oracle Web Cache caches full-page dynamic content.

To cache the `snoop` servlet:

1. Ensure that Oracle Web Cache has been configured to communicate with the WebSphere Application Server, as described in Section 12.1.1.

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 Section 6.8.

   When creating the caching rule for the `snoop` output, ensure you configure the following in the Create Caching Rule page:

   - Click the **Cache** check box.
   - In the **Match URL By** section, select **Path Prefix** and enter `/snoop`.
   - In the **HTTP Methods** section, click **GET.**

4. Point the browser to Oracle Web Cache with following URL:

   `http://`*`web_cache_hostname`*`:`*`admin_port`*`/snoop`

   See Section 2.11.1.1 to determine the port.

   The output is the same when you accessed `snoop` directly from the WebSphere Application Server. This time, Oracle Web Cache caches the `snoop` output and serves the response to the browser.

5. View the contents of the cache, as described in Section 8.6, 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.

12.2.2 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, install Technology Samples, as mentioned in the documentation under IBM WebSphere Application Server samples gallery. Use this JSP to demonstrate how Oracle 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 Section 6.7.

   In the Create Expiration Policy dialog box, perform the following steps:

   a. In the **Objects Expire** section, select **After Cache Entry** and enter `60` seconds in the **Time Limit** field.

   b. In the **Action On Expired Objects** section, select **Remove Immediately.**

3. Create a session caching rule, as described in Section 6.8.6.

   When configuring a session caching rule, perform the following steps:

   a. When creating a session definition in the **Session Definitions** section of the Session Configuration page:

      * In the **Session Name** field, enter `IBMSession`.
      * In the **Cookie Name** field, enter `JSESSIONID`.
      * In the **URL Post Body Parameters** field, enter `jsessionid`.

   b. In the **Session Policy Configuration** section of the Session Configuration page, create two policies named `IBMSession`:

      * In the **Cache** column for the first **IBMSession** policy, select the **With Session** option.
      * In the **Cache** column for the second **IBMSession** policy, select the **Without Session** option.
      * Do not select the **Substitute Default Value** check box.

   c. Create a new caching rule for `Calendar`.

      When creating the caching rule for the `Calendar` output, configure the following in the **General** tab of the Create Caching Rule page:

* Click the **Cache** check box.
* From the **Expiration** list, select **Expire 60 seconds after cache entry and remove immediately**.
* In the **Match URL By** section, select **Path Prefix** and enter `/TechnologySamples/Calendar`.
* In the **HTTP Methods** section, click **GET.**

In the **Sessions** tab of Create Caching Rule page, select both the **IBMSession** sessions, one using setting **Without Session** and the other using setting **With Session**.

4. Point the browser to Oracle Web Cache with the following URL:

   `http://`*`web_cache_hostname`*`:`*`WebCache-admin_port`*`/TechnologySamples/Calendar`

   See Section 2.11.1.1 to determine the port.

   The output is the same when you accessed `Calendar` directly from WebSphere Application Server. This time, Oracle Web Cache caches the `Calendar` output.

5. View the contents of the cache, as described in Section 8.6, 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, Oracle Web Cache expires the cached content after 60 seconds and reflects the content the next time the user requests the page.

After deploying Oracle 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="*:WebCache-admin_port"/>`.
2. In the WebSphere Application Server administrative console, navigate to **Environment** > **Virtual Hosts**.
3. Follow the prompts to add a new virtual host.

12.3 Apache Tomcat

Apache Tomcat, Version 4.1 is a servlet container. It is included with the Apache Jakarta Project. The Apache Tomcat installation includes several JSP and Java servlet examples. This section explains how to configure Oracle Web Cache to cache the following content:

- Section 12.3.1, "Apache Tomcat Snoop JSP"
- Section 12.3.2, "Apache Tomcat Session Servlet"

Follow the instructions enclosed within the Apache Tomcat binary for installation. Apache Tomcat requires the Java Development Kit (JDK).

For more information, see:

- `http://jakarta.apache.org/tomcat/` for further information about Apache Tomcat
- `http://java.sun.com` for further information about downloading and installing JDK

12.3.1 Apache Tomcat Snoop JSP

`snoop.jsp` shows getting and using request information, headers, and parameters sent by the browser. Use it to demonstrate how Oracle Web Cache caches full-page dynamic content.

To start, perform the following steps:

1. Ensure that Oracle Web Cache has been configured to communicate with the Apache Tomcat server, as described in Section 12.1.1.
2. Start the Apache Tomcat server, and then access the following URL:

   `http://`*`web_cache_hostname`*`:`*`WebCache-admin_port`*`/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 Section 6.8.

   When creating the caching rule for the `snoop` output, ensure you configure the following in the Create Caching Rule page:

   - Click the **Cache** check box.
   - In the **Match URL By** section, select **Path Prefix** and enter `/examples/jsp/snp/snoop.jsp`.
   - In the **HTTP Methods** section, click **GET.**
2. Point the browser to the Oracle Web Cache with following URL:

   `http://`*`web_cache_hostname`*`:`*`admin_port`*`/eexamples/jsp/snp/snoop.jsp`

   See Section 2.11.1.1 to determine the port.

   The output is the same when you accessed `snoop` directly from Apache Tomcat. This time, Oracle Web Cache caches the `snoop` output and serves the response to the browser.
3. View the contents of the cache, as described in Section 8.6, 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.

12.3.2 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 Oracle 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 12–1.

Example 12–1 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 Oracle Web Cache has been configured to communicate with the Apache Tomcat, as described in Section 12.1.1.
4. Configure the browser not to accept cookies.

   This is required 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 Section 6.7.

   In the Create Expiration Policy dialog, perform the following steps:

   a. In the **Objects Expire** section, select **After Cache Entry** and enter `60` in the **Time Limit** field.

b. In the **Action on Expired Objects** section, select **Remove Immediately.**

2. Create a session caching rule, as described in Section 6.8.6.

   When configuring a session caching rule, perform the following steps:

   a. When creating a session definition in the **Session Definitions** section of the Session Configuration page:

      * In the **Session Name** field, enter `ApacheSession`.
      * In the **Cookie Name** field, enter `JSESSION`.
      * In the **URL Post Body Parameters** field, enter `jsessionid`.

   b. In the **Session Policy Configuration** section of the Session Configuration page, create two policies named `ApacheSession`:

      * In the **Cache** column for the first **ApacheSession** policy, select the **With Session** option.
      * In the **Cache** column for the second **ApacheSession** policy, select the **Without Session** option.
      * Do not select the **Substitute Default Value** check box.

   c. Create a new caching rule for `Session`.

      When creating the caching rule for the `SessionServlet` servlet output, configure the following in the **General** tab of the Create Caching Rule page:

      * Click the **Cache** check box.
      * From the **Expiration** list, select **Expire 60 seconds after cache entry and remove immediately**.
      * In the **Match URL By** section, select **Path Prefix** and enter `/examples/servlets/SessionServlet`.
      * In the **HTTP Methods** section, click **GET.**

      In the **Sessions** tab of Create Caching Rule page, select both the **ApacheSession** sessions, one using setting **Without Session** and the other using setting **With Session**.

3. Create a session caching rule as described in Section 6.8.6.

4. Point the browser to Oracle Web Cache with the following URL:

   `http://`*`web_cache_hostname`*`:`*`WebCache-admin`*`/examples/servlets/SessionServlet`

   See Section 2.11.1.1 to determine the port.

   The output is the same when you accessed `Session` servlet directly from Apache Tomcat. This time Oracle 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 Oracle Web Cache serves the content, and the request never goes to the Apache Tomcat.

5. View the contents of the cache, as described in Section 8.6, 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, Oracle Web Cache expires the cached content after 60 seconds and reflects the content the next time the user requests the page.

12.4 Microsoft IIS

The Microsoft IIS installation includes several ASP examples. This section explains how to configure Oracle Web Cache to cache the following content:

- Section 12.4.1, "ServerVariables_Jscript ASP"
- Section 12.4.2, "Cookie_Jscript ASP"

12.4.1 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 Oracle Web Cache caches full-page dynamic content.

To start, perform the following steps:

1. Ensure that Oracle Web Cache has been configured to communicate with IIS, as described Section 12.1.1.
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 Section 6.8, using the following information:, configure the following in the Create Caching Rule page:

   When creating the caching rule for the ServerVariables_JScript.asp output, configure the following in the Create Caching Rule page:

   - In the **Match URL By** section, enter:

     `/IISSamples/sdk/asp/interaction/ServerVariables_JScript.asp`

   - In the **HTTP Methods** section, click **GET.**
   - In the **Caching Response** section, click **Cache.**
2. Point the browser to Oracle Web Cache with following URL:

   `http://`*`web_cache_hostname`*`:`*`WebCache-admin_port`*`/eIISSamples/sdk/asp/interaction/ServerVariables_JScript.asp`

   See Section 2.11.1.1 to determine the port.

   The output is the same when you accessed `ServerVariables_JScript.asp` directly from IIS. This time, Oracle Web Cache caches the `ServerVariables_JScript.asp` output and serves the request to the browser.
3. View the contents of the cache, as described in Section 8.6, 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.

12.4.2 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 Oracle Web Cache caches pages with session cookies.

To start, perform the following steps:

1. Ensure that Oracle Web Cache has been configured to communicate with IIS, as described in Section 12.1.1.
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 Section 6.7.

   In the Create Expiration Policy dialog, perform the following steps:

   a. In the **Objects Expire** section, select **After Cache Entry** and enter `60` in the **Time Limit** field.

   b. In the **Action on Expired Objects** section, select **Remove Immediately.**

2. Create a session caching rule, as described in Section 6.8.6.

   When configuring a session caching rule, perform the following steps:

   a. When creating a session definition in the **Session Definitions** section of the Session Configuration page:

      * In the **Session Name** field, enter `MSSession`.
      * In the **Cookie Name** field, enter `CookieJSCript`.
      * In the **URL Post Body Parameters** field, enter `jsessionid`.

   b. In the **Session Policy Configuration** section of the Session Configuration page, create two policies named `MSSession`:

      * In the **Cache** column for the first **MSSession** policy, select the **With Session** option.
      * In the **Cache** column for the second **MSSession** policy, select the **Without Session** option.
      * Do not select the **Substitute Default Value** check box.

   c. Create a new caching rule for `Cookie_JScript.asp`.

      When creating the caching rule for the `Cookie_JScript.asp` output, configure the following in the **General** tab of the Create Caching Rule page:

      * Click the **Cache** check box.
      * From the **Expiration** list, select **Expire 60 seconds after cache entry and remove immediately**.
      * In the **Match URL By** section, enter `/IISSamples/sdk/asp/interaction/Cookie_JScript.asp`.
      * In the **HTTP Methods** section, click **GET.**

In the **Sessions** tab of Create Caching Rule page, select both the **MSSSession** sessions, one using setting **Without Session** and the other using setting **With Session**.

3. Point the browser to Oracle Web Cache with the following URL:

   ```
   http://web_cache_hostname:WebCache-admin_
   port/eIISSamples/sdk/asp/interaction/Cookie_JScript.asp
   ```

   See Section 2.11.1.1 to determine the port.

   The output is the same when you accessed `Cookie_JScript.asp` directly from IIS. This time, Oracle 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 Oracle Web Cache serves the cached content, and the request never goes to IIS.

4. View the contents of the cache, as described in Section 8.6, 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, Oracle Web Cache expires the cached content after 60 seconds and reflects the content the next time the user requests the page.

A

Troubleshooting Oracle Web Cache

This appendix describes common problems that you might encounter when using Oracle Web Cache and explains how to solve them. It contains the following topics:

A.1 Problems and Solutions

This section describes common problems and solutions. It contains the following topics:

A.1.1 No Response from Application Web Server Error

Problem

If an 11*g* Release 1 (11.1.1) Oracle Web Cache is reverse proxying 10*g* components, such as Oracle Portal, Oracle Forms Services, or Oracle Business Intelligence Discoverer, and SSL is enabled, the following browser error may return:

```
No response from Application Web Server
```

In addition, messages similar to the following appear in the event log:

```
[2009-02-11T03:44:55-08:00] [webcache] [ERROR:32] [WXE-11904] [security]
[ecid: ] SSL handshake fails NZE-29024
[2009-02-11T03:44:55-08:00] [webcache] [WARNING:1] [WXE-11905] [security]
[ecid: ] SSL additional information: The certificate sent by the other side could
not be validated. This may occur if the certificate has expired, has been revoked,
or is invalid for another reason.
[2009-02-11T03:44:55-08:00] [webcache] [WARNING:1] [WXE-11905] [security]
[ecid: ] SSL additional information: Remote IP [140.87.11.159]:9002
[2009-02-11T03:44:55-08:00] [webcache] [WARNING:1] [WXE-11905] [security]
[ecid: ] SSL additional information: Local IP 152.68.64.152:2832
[2009-02-11T03:44:55-08:00] [webcache] [WARNING:1] [WXE-11906] [security]
[ecid: ] SSL details: SSL error during handshake (details: internal=The
certificate sent by the other side could not be validated. This may occur if the
certificate has expired, has been revoked, or is invalid for another reason.
system=Success)
```

This error indicates that the wallet you selected for Oracle Web Cache contains a certificate that does not match the wallet used by the 10*g* components.

Solution

To resolve this problem, modify the wallet you specify for Oracle Web Cache to use the wallet you are using for the 10*g* components. See Section 5.4.3.

A.1.2 Load Issues on Oracle Web Cache Computer

On UNIX operating systems, the `top` and `uptime` utilities report a higher than expected average load when the Oracle Web Cache computer is idle.

Problem

This effect occurs because Oracle Web Cache performs light maintenance work, even when it is idle. One operation Oracle 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. Oracle Web Cache increases the average queue length (uptime output) by approximately one.
- The CPU load is still low because the work Oracle Web Cache performs is minimal.

A.1.3 Performance Degradation and Memory

Because Oracle Web Cache is an in-memory cache, it is best to deploy Oracle Web Cache on a dedicated computer to minimize paging. Unless the computer is dedicated to run Oracle 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: Oracle 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 Oracle Web Cache to work efficiently on a computer with paging, either deploy Oracle Web Cache on a dedicated computer or reduce the maximum cache size and maximum cached object size. See Section 2.11.5 to configure these settings.

Problem 2: Oracle Web Cache Using Memory than the Maximum Cache Size

If Oracle 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, Oracle 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. See Section 2.11.5 to configure these settings.

A.1.4 Invalidation Timeouts in a Cache Cluster

Invalidation has a default timeout of 300 seconds for the propagation of invalidation requests in the **cache hierarchy** or **cache cluster** deployments. See Section 7.10.2 for an overview of invalidation propagation.

Problem

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

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 run in a LAN environment. Therefore, decreasing the value of `INV_PEER_TIMEOUT` typically improves efficiency.

4. Save `webcache.xml`.
5. Restart Oracle Web Cache with the following command:

   ```
   opmnctl restartproc ias-component=component_name
   ```

   This executable is found in the following directory:

   ```
   (UNIX) ORACLE_INSTANCE/bin
   (Windows) ORACLE_INSTANCE\bin
   ```

6. Synchronize configuration changes to all cache cluster members and restart the other cache cluster members. See Section 3.6.5 to perform this task in Fusion Middleware Control and Section 3.7.4 to perform this task in Oracle Web Cache Manager.

A.1.5 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.
```

In addition, messages similar to the following appear in the event log:

```
[2009-04-16T11:56:23-04:00] [webcache] [WARNING:1] [WXE-14021] [backend] [ecid:
453611135182,0] Capacity, as set by Web Cache, is exceeded for all origin servers
that are available for processing this request.
[2009-04-16T11:56:23-04:00] [webcache] [ERROR:32] [WXE-11365] [frontend] [ecid:
453611135182,0] Server busy response is returned.
```

These errors indicate that the application Web server has exceeded the maximum number of concurrent connections.

Solution

To resolve this problem, you can either:

- Increase capacity. See Section 2.11.2.
- Evaluate the caching rules to determine if additional content can be cached. See Section A.3.

A.1.6 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 Oracle Web Cache, the browser expects more data to arrive and the connection does eventually time out. If the actual length of the page is greater than the `Content-Length`, the browser does not receive the complete page. This problem does not occur for cache hits because Oracle Web Cache correctly calculates the content length itself for pages stored in the cache.

Messages similar to the following appear in the event log:

```
[2009-04-16T13:23:32-04:00] [webcache] [WARNING:1] [WXE-11093] [backend] [ecid:
268932876536,0] Content-Length value 7755 sent by
origin server does not match number of bytes received, that is 454 bytes for
localhost:7785/cgi-bin/test-cgi.
```

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.

A.1.7 Browser Presenting 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 Oracle Web Cache configuration conditions, may receive an error saying that the page cannot be displayed:

- Oracle 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.

Please see the following related sections:

- Section 5.4 for further information about configuring Oracle Web Cache to accept HTTPS requests
- Section 2.11.5 for further information about configuring the keep-alive timeout
- Section 5.4.4.1 for further information about configuring the Oracle HTTP Server to maintain keep-alive connections from Oracle Web Cache for Internet Explorer browsers

When Oracle Web Cache is configured with these settings, Internet Explorer may send HTTPS requests *after* Oracle 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, Oracle 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 Oracle Web Cache with the following command:

   ```
   opmnctl restartproc ias-component=component_name
   ```

   This executable is found in the following directory:

   ```
   (UNIX) ORACLE_INSTANCE/bin
   (Windows) ORACLE_INSTANCE\bin
   ```

A.1.8 ESI Errors with IBM Websphere Application Server

Due to an incompatibility with the ESI capability token between IBM Websphere Application server and Oracle Web Cache, you may see errors.

Problem

If Oracle Web Cache is deployed to cache content for an IBM Websphere Application server, the following errors may result:

- Error page with text "Some ESI Features in this page are not supported"
- Images do not render

Solution

To resolve this issue

1. In the WebSphere Application Server administrative console, navigate to **Server**s > **Application Servers**.
2. In the Application servers page, 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 an entry.
6. In the **Name** field, enter `com.ibm.servlet.file.esi.control`.
7. In the **Value** field, enter the following string:

   ```
   max-age=300, cacheid="URL",content="ESI/1.0".
   ```

8. Click **Apply** to save the settings.
9. Restart IBM Websphere Application Server.

A.1.9 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.

A.2 Common Configuration Mistakes

Common configuration mistakes include:

- Not mapping sites correctly to origin servers.

  When sites are not mapped, Oracle 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 Section 2.11.3 for further information about configuring sites.

- Ping URL

  When configuring the ping URL, 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 Section 2.11.2 for further information about configuring origin servers.

- Running Oracle Web Cache with root privilege

  On UNIX, you must configure Oracle Web Cache to run with root privilege in the following cases:

  - Privileged port numbers less than 1024 are being used for Oracle Web Cache listening ports.
  - There are more than 1,024 file descriptors being used for connections to Oracle Web Cache.
  - The current `opmnctl` user does not match the configured process identity user.

  See Section 5.9 for further information about configuring origin servers.

The TRACE verbosity event-logging level can help you validate Oracle Web Cache configuration settings, such as:

- Site resolution
- Site-to-server mappings route to the correct origin servers
- Compression
- Session binding
- Caching rules
- ESI processing

See Section 9.3 to configure TRACE verbosity in the event logs.

A.3 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. See Section 8.6.
   - Listing the contents of the cache. See Section 8.6.
   - Previewing invalidation without invalidating actual content. See Section 7.7.2 to use the **Preview list of objects that match invalidation criteria** option.
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 Section 9.3 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.

   See Section 6.6 for further information about configuring caching rules.
4. Enable access logging. Then, send an explicit request for the object.

   By analyzing the access log determine, you can determine if Oracle Web Cache is serving the object from its cache or is forwarding the request to the origin server.

   See Section 9.4 for further information about enabling access logging.

A.4 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, as described in Section 9.3. It is also useful to display the diagnostic information and event log information in the HTML response body, as described in Section 8.3.

The following topics describe using the event log and HTML response body to diagnose template and fragment syntax errors:

- Section A.4.1, "Template Syntax Error Example"
- Section A.4.2, "Fragment Syntax Error Example"
- Section A.4.3, "Fragment Syntax Error with Exception Handling Example"

A.4.1 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
```

A.4.2 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.
```

A.4.3 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.
```

A.5 Impact of HTTP Traffic Changes

When Oracle 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 Oracle Web Cache, configure all absolute URLs to use the protocol, host name, and port number of Oracle Web Cache. Also, ensure the `Port` directive in the Oracle HTTP Server `httpd.conf` file specifies the Oracle Web Cache listening port.

- SSL Processing

  Add certificate management to Oracle Web Cache, if the connection between the client and Oracle Web Cache is HTTPS, but the connection between Oracle Web Cache and the origin server is HTTP.

- Page Delivery Timing

  For compressed pages or pages that requires processing, Oracle Web Cache waits for an entire page from the origin server before it sends it to the browser.

- HTTP Protocol

  Oracle Web Cache transparently performs the following:

  - Oracle Web Cache upgrades and downgrades the protocol version between the origin server and client.
  - For cacheable objects, Oracle Web Cache sends content to clients with the `Content-Length` response header instead of chunked encoding for the initial request.

- For cache hits, Oracle 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.

A.6 Need More Help?

You can find more solutions at the following locations:

- Oracle *MetaLink*, `http://metalink.oracle.com`. If you do not find a solution for your problem, log a service request.
- Oracle Web Cache Forum:

  `http://forums.oracle.com/forums/forum.jspa?forumID=8`

Glossary

access log

A log file that contains information about the HTTP requests sent to Oracle Web Cache for a Web site. The access log has a file name of `access_log` and is stored by default in the following directories:

`(UNIX)` *`ORACLE_INSTANCE`*`/diagnostics/logs/WebCache/<`*`webcache_name`*`>`
`(Windows)` *`ORACLE_INSTANCE`*`\diagnostics\logs\WebCache\<`*`webcache_name`*`>`

admin server process

An Oracle 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.

cache cluster

A loosely coupled collection of cooperating Oracle Web Cache cache instances to provide a single logical cache. Cache clusters provide failure detection and failover of caches, increasing the availability of your Web site.

cache cluster member

An instance of Oracle Web Cache configured with other instances of Oracle 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 Oracle Web Cache caches content from another Oracle Web Cache to a local market. Oracle 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 Oracle 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 Oracle 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 Oracle 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**.

DMZ

A demilitarized zone (DMZ) or perimeter network is a network area (a subnetwork) that sits between an organization's internal network and an external network, usually the Internet. The point of a DMZ is that connections from the internal and the external network to the DMZ are permitted, whereas connections from the DMZ are only permitted to the external network; hosts in the DMZ may not connect to the internal network. This allows the DMZ's hosts to provide services to both the internal and external network while protecting the internal network in case intruders compromise a host in the DMZ.

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. Oracle Web Cache accepts requests that use the following characters as delimiters: question mark (`?`), ampersand (`&`), dollar sign (`$`), or semicolon (`;`).

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 Oracle 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 Oracle Web Cache event and error information. The event log has a file name of `event_log` and is stored in the following directories:

```
(UNIX) ORACLE_INSTANCE/diagnostics/logs/WebCache/<webcache_name>
(Windows) ORACLE_INSTANCE\diagnostics\logs\WebCache\<webcache_name>
```

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.

farm

A collection of components managed by Fusion Middleware Control. It can contain zero or one managed server domains and the Oracle Fusion Middleware system components that are installed, configured, and running on the domain.

failover

When an **origin server** fails, Oracle 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, Oracle Web Cache

transfers ownership of the content of the failing member to the remaining cluster members.

failure detection

In a cache cluster environment, Oracle 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 Oracle 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 Oracle 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 Oracle Web Cache cache consistent with the content on the origin servers.

invalidation coordinator

In a cache cluster environment, Oracle 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 Oracle Web Cache server. Oracle 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.

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 Oracle Web Cache
- View performance metrics

Oracle Web Cache Manager

A tool that combines configuration abilities with component control to provide an integrated environment for configuring and managing Oracle 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. Oracle 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 Oracle 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 Oracle Web Cache to fetch and include the HTML fragments. The fragments themselves are HTML files containing discrete text or other objects.

performance assurance heuristics

Heuristics that enable Oracle Web Cache to assign a queue order to 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 Oracle 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 in one central location and accessed through a common interface, called a page.

provider cache

In an **ESI cache hierarchy**, an Oracle 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

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

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

remote cache

In a **distributed cache hierarchy**, an Oracle 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 back-end 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, Oracle Web Cache can cache pages for **HTTPS protocol** requests.

selectors

Oracle 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**

session binding

The process of binding a user session to a given **origin server** to maintain state for a period.

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 Oracle 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 Oracle 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 travels 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 Oracle Web Cache. Browsers can request cached content from these sites through Oracle Web Cache. In addition to caching content, Oracle 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 Oracle Web Cache is running in a standalone environment (that is, you installed Oracle Web Cache from a kit that included only this product; you did not install Oracle Web Cache as part of an Oracle Application Server installation).

XLF

See **Extended Log Format (XLF)**.

XML

See **Extensible Markup Language (XML)**.

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X