Caching (Static & Dynamic)
Deployment Guide
A Step-by-Step Technical Guide
# Table of Contents

Introduction ........................................................................................................... 4
Solution Requirements .............................................................................................. 5
Prerequisites ............................................................................................................. 5
Network Diagram ..................................................................................................... 6
First time connectivity ............................................................................................ 7
  Serial Connection ................................................................................................ 7
  Ethernet Connection ............................................................................................ 7
NetScaler Configuration .......................................................................................... 8
  Deployment Model: Netscaler High Availability, Two-Arm Mode, Static and Dynamic Caching .................................................................................................. 8
About the Integrated Cache .................................................................................... 9
  Basic Information ................................................................................................. 9
  Components of the Cache .................................................................................... 9
  Policies - Pre-Default, Default, Post-Default ...................................................... 10
  Important Policy Behavior - Policy Engine (PE) ................................................ 10
  Enabling Caching ............................................................................................... 12
  Important Cache Terminology ........................................................................... 12
  Things that are automatically cached ............................................................... 13
  Things that are not cached ................................................................................ 13
  304-Hits ............................................................................................................ 14
  Expiration Model ............................................................................................... 15
Application Profiling .............................................................................................. 16
  Taking a trace ..................................................................................................... 16
  Taking a trace with wireshark ........................................................................... 16
  Taking a trace with the Citrix Application Switch ............................................. 16
  Taking a trace with the Citrix Application Switch ............................................. 17
  Viewing headers with Paros ............................................................................... 17
  Viewing headers with Live HTTP Headers ....................................................... 17
  Viewing headers with IE Analyzer .................................................................... 17
  Viewing headers with IE Watch ........................................................................ 17
Caching - Static ..................................................................................................... 18
  Static Caching ................................................................................................... 18
  What is not cacheable ....................................................................................... 18
  Citrix built-in no-cache policies ....................................................................... 19
  Custom no-cache policies ................................................................................. 19
  What is cacheable .............................................................................................. 20
  Citrix built-in cache policies - placed into DEFAULT group .............................. 20
  Custom cache policies - common cache policies .......................................... 20
Caching - Dynamic ............................................................................................... 22
  Dynamic Caching ............................................................................................. 22
  Dynamic Cache Selectors .................................................................................. 22
  Oracle Navigator ............................................................................................... 24
  Oracle Navigator selectors - Hit group 1 ......................................................... 24
  Oracle Navigator selectors - Hit group 2 ......................................................... 24
  Oracle Navigator selectors - Invalidation group ............................................. 24
  Oracle Forms ..................................................................................................... 28
  Oracle Forms selectors - Hit group ................................................................. 29
  Oracle Forms selectors - Invalidation group ................................................... 29
  Oracle Reporting .............................................................................................. 30
  Oracle Logout Invalidation ............................................................................... 32
  Oracle Logout selectors - Invalidation group .................................................. 33
Appendix A - NetScaler Application Switch Configuration .................................. 34
Introduction

Citrix® NetScaler® optimizes the delivery of web applications — increasing security and improving performance and Web server capacity. This approach ensures the best total cost of ownership (TCO), security, availability, and performance for Web applications. The Citrix NetScaler solution is a comprehensive network system that combines high-speed load balancing and content switching with state-of-the-art application acceleration, layer 4-7 traffic management, data compression, dynamic content caching, SSL acceleration, network optimization, and robust application security into a single, tightly integrated solution. Deployed in front of application servers, the system significantly reduces processing overhead on application and database servers, reducing hardware and bandwidth costs.

The Integrated Caching feature of the Citrix Application Switch helps optimize the delivery of web content and applications. The Integrated Cache is a fast, in-memory cache that is HTTP/1.1 and HTTP/1.0 compliant by default, but can be configured to do much more. The Integrated Cache can store a variety of static and dynamic content and serve this content instantly to many users. At the origin (for example, data center deployments fronting the web servers), aside from the inherent benefits of the system, caching of HTTP content reduces the number of web server transactions.

Caching dynamic content reduces the latency and the computation cost (in the application and database servers) associated with the dynamic page generation process. Caching at the edge (for example, ISP and branch Enterprise deployments) can, in addition, result in significant reduction in page download time and in bandwidth utilization.

This deployment guide was created as the result of validation testing with the Oracle Enterprise Business Suite v12 application. This deployment guide walks through the step-by-step configuration details of how to configure the Citrix NetScaler application switch for Caching and some of the considerations necessary for integration with Oracle EBSv12.
Solution Requirements

- Application Delivery Front-End
  - Static Caching
  - Dynamic Caching
  - Oracle E-Business Suite v12

Prerequisites

- Citrix NetScaler L4/7 Application Switch, running version 8.0+, (Quantity x 1 for single deployment, Quantity x 2 for HA deployment).
- Layer 2/3 switch, w/support for 802.1q VLANs, (Quantity x 1)
- Client laptop/workstation running Internet Explorer 6.0+, Ethernet port
- 9-pin serial cable -or- USB-to-serial cable

NOTE:
The policies in this guide are based on the Policy Engine (PE) architecture in NetScaler version 8.0. The policies for NetScaler version 9.0+ use the Policy Infrastructure (PI) architecture which are different in syntax and methodology. Policy Infrastructure is not discussed in this guide.
# Network Diagram

The following is the Network that was used to develop this deployment guide, and is representative of a solution implemented at a customer site.

<table>
<thead>
<tr>
<th>VLAN Legend</th>
<th>Primary NetScaler</th>
<th>Primary/Secondary NetScaler</th>
<th>Secondary NetScaler</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLAN 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VLAN 10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VLAN 11</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**VLAN 1**
- IP Addresses: NSIP: 10.217.104.51 / 24

**VLAN 10**
- Interface 1/2, Untagged

**VLAN 11**
- Interface 1/5, Untagged
- MIP: 67.97.253.84 / 29

**VLAN 1 (Mgmt)**
- Interface 0/1, Untagged
- SNIP: 10.217.104.54 / 24

**IP Addresses**
- NSIP: 10.217.104.52 / 24

**Shared IP Addresses**
- VIP: 67.97.253.91 / 29
- VIP: 67.97.253.92 / 29

**Oracle Database Server**
- 169.145.91.81

**Oracle Application Server**
- 169.145.91.82
First time connectivity

Serial Connection

The NetScaler can be accessed by the serial port through any terminal emulation program. Windows Hyperterm is commonly used on a laptop or workstation. Connect a 9-pin Null Modem cable (or USB-to-9-pin cable) from the computer to the NetScaler’s console port. In the terminal emulation program configure the settings for 9600 baud, No stop bits, 8 data bits, and 1 parity bit. The login prompt should appear. The default login is nsroot, nsroot. It is advisable to change the nsroot password once connected.

Once connected type in the CLI command ‘configns’ (‘nsconfig’ if at the shell prompt). Select option 1 to change the NetScaler IP Address and Network Mask. Exit, save and reboot.

Ethernet Connection

The NetScaler can also be accessed by the default IP Address of 192.168.100.1, either through an http, https, telnet or ssh connection. Once connected, the login prompt should appear. The default login is nsroot, nsroot. It is advisable to change the nsroot password once connected.

Type in the CLI command ‘configns’ (‘nsconfig’ if at the shell prompt). Select option 1 to change the NetScaler IP Address and Network Mask. Exit, save and reboot.

Note: Changing the NetScaler IP Address always requires a reboot.
NetScaler Configuration

Deployment Model: Netscaler High Availability, Two-Arm Mode, Static and Dynamic Caching.

The NetScalers in this example assume a high availability pair configuration, in two-arm mode. All configuration changes will be made on the Primary NetScaler and will be propagated to the Secondary NetScaler. The NetScalers in Two-Arm mode provide the utmost in site performance, as the NetScaler evaluates integrated caching policies and stores cacheable objects, while non-cacheable traffic is sent to the backend application servers.

Connect to the NetScaler via the NSIP using a web browser.

In this example:
NS1: http://10.217.104.51
NS2: http://10.217.104.52

Note: Java will be installed.

Default login is: nsroot, nsroot.
About the Integrated Cache

Basic Information

The Integrated Cache in the NetScaler is HTTP 1.1, HTTP 1.0 and RFC 2616 compliant. If you are not familiar with RFC 2616, now is a good time to become familiar with Caching as it relates to RFC 2616, http://www.ietf.org/rfc/rfc2616.txt. The Integrated Cache in the NetScaler is a fast, in-memory store that serves responses to HTTP requests without requiring a round trip to an origin server. The Integrated Cache can be configured to store and serve static as well as dynamic content.

The Installation and Configuration Guide, Volume 2 (NS_ICG_V2.pdf) provides more detail surrounding Integrated Caching configuration and should be used as another reference.

- Only HTTP and HTTPS transactions are cached. Caching of HTTPS transactions is only possible if the system is also doing SSL encryption on behalf of the origin web servers.

- Both Static and Dynamic content can be cached. To ensure the freshness of cached content, a variety of expiration and invalidation mechanisms are provided. An important rule of thumb to remember is, “If you cache, you must expire”.

- Content amounting to half of the available memory (RAM) in the NetScaler can be cached.

Components of the Cache

Content Group

The NetScaler stores cached objects in content groups. By default, static objects are stored in the DEFAULT content group. You can define new content groups for static and dynamic cache objects.

Selectors

A Selector is part of a content group, that matches particular strings, parameterized data, query strings, cookies and POST data. Selectors define expressions based on strings or other values in an HTTP request, and are then used to locate that data for serving it or marking it obsolete or stale. A static string in a selector can be a URL stem, a parameter that appears in the request, a static string in a POST body, and so on. Dynamic parameters can be based on variable data in HTTP headers, cookies, or the body of the response. For example, when a user clicks the Submit button in an HTML form, the resulting request contains parameters that represent fields on the form. Each form that is submitted will have unique values associated with the field parameters, for example, a Name field will have a different value every time the form is submitted. You can define a selector that looks at various parameter-value pairs, for example, NameField=username and Method=POST, and finds matching items based on both the static method string and the dynamic user name data. A different set of cache objects would be stored for NameField=“TIM” and for NameField=“JOHN”, for example.

Policies

Policies enable the NetScaler to determine what objects to cache and what content group to store them in. There are built-in default policies for HTTP requests and responses, and they only work with static content and the DEFAULT content group.
Policies - Pre-Default, Default, Post-Default

There exist two sets of built-in cacheability policies - one for request time processing and another for response time. These built-in policies encode the standard HTTP behavior which is part of the default behavior of the Integrated Cache.

These policies are hard-coded for efficiency and will not show up in the show cache policy command. For more information these can be found in the ICG Volume 2, Appendix C, “Understanding Cache Policies,” on page C-1.

You can bind a policy to be evaluated before or after the built-in policies. To do so, in the bind cache global command, set -precedeDefRules to YES or NO. This leads to one more classification of Integrated Cache policies: Pre-Default, Default, or Post-Default.

It is important to understand the default behavior, so that you can decide whether the policy should be pre-default or post-default. Since pre-default policies are evaluated before the built-in policies, they override the default behavior. Post-default policies add to the default behavior. If you wish, you can fully control the behavior of the Integrated Cache by providing all the controls in the pre-default policies and making sure that the built-in policies are never evaluated.

Important Policy Behavior - Policy Engine (PE)

Policies get evaluated in the order that they are classified in, that is with their priority numbers. Policies operate on a first-match principle. In a policy classification, the action associated with the first policy that matches gets applied. Once a match is determined, the policy evaluation exits the evaluation logic tree. If a request matches a invalidation policy then that request will bypass the cache.

If Bypass Cache is set to Enabled (the default), the hit evaluation only happens after the pre-default, default, and post-default request cacheability policies have been evaluated.
If Bypass Cache is Disabled, the hit evaluation happens before the pre-default, default, and post-default request cacheability policies have been evaluated. Disabling Bypass will lead to better performance, because most hits will be served without any cacheability policy evaluation. Cacheability evaluation only happens for requests that don’t find a matching object in the cache, that is, for cache misses.

**NOTE:**

The first CACHE or NOCACHE cacheability policy that matches decides what happens to the transaction, that is, whether the response object is cacheable, and which group the object should be part of. So, if the cacheability decision has already been made at request time, then no further cacheability policy valuation will take place. This behavior follows the architecture of Policy Engine (PE) in NetScaler versions leading up to v8.0. This behavior will be more flexible in Policy Infrastructure (PI) in NetScaler versions v9.0+.
Enabling Caching

The NetScaler should have the appropriate license installed to enable Integrated Caching.

![Configure Basic Features]

Navigate to System ➔ Settings ➔ Basic Features. Select Integrated Caching and click OK.

Important Cache Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache Hit</td>
<td>The cache engine finds the object in the cache, and serves it to the client.</td>
</tr>
<tr>
<td>Cache Miss</td>
<td>The cache engine does not find the object in the cache, and it fetches the object from the backend, origin server and servers it to the client. A storable cache miss is stored in the cache, a non-storable cache miss is not stored in the cache according to policy.</td>
</tr>
<tr>
<td>Expiration</td>
<td>If the cached object has expired, the caching engine fetches a new version from the backend server (origin server) and stores the updated object in the cache.</td>
</tr>
<tr>
<td>Prefetch</td>
<td>Refreshes an object before it expires so that it never becomes stale before it is requested again. Improves performance.</td>
</tr>
<tr>
<td>Cache Buffering</td>
<td>Serves an object as soon as it receives the response header, rather than waiting for the entire response to be downloaded.</td>
</tr>
<tr>
<td>Flash Cache</td>
<td>Only sends one request to the backend server when a flood of requests arrive at the NetScaler (Flash Crowd). *CAUTION, read the manual before turning this on. All requests are queued and whatever response is sent from the server, is sent to all queued clients.</td>
</tr>
<tr>
<td>Poll Every Time (PET)</td>
<td>Every request for a Poll Every Time object requires revalidation. Typically, the PET feature is enabled on objects that are not modified often, but do need revalidation. The PET feature is beneficial when there is a large percentage of successful revalidations.</td>
</tr>
<tr>
<td>Things that are automatically cached</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---</td>
</tr>
<tr>
<td><strong>Object</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>All static objects</td>
<td>Image files, html pages, css, js, jar</td>
</tr>
<tr>
<td>200</td>
<td>OK pages</td>
</tr>
<tr>
<td>203</td>
<td>Non-Authoritative Response</td>
</tr>
<tr>
<td>300</td>
<td>Multiple Choices pages</td>
</tr>
<tr>
<td>301</td>
<td>Moved Permanently pages</td>
</tr>
<tr>
<td>302</td>
<td>Found pages</td>
</tr>
<tr>
<td>304</td>
<td>Not Modified pages</td>
</tr>
<tr>
<td>307</td>
<td>Temporary Redirect pages</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden pages</td>
</tr>
<tr>
<td>404</td>
<td>Not Found pages</td>
</tr>
<tr>
<td>410</td>
<td>Gone pages</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Things that are not cached</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>2xx</td>
<td>201, 202, 204, 205, 206 status codes</td>
</tr>
<tr>
<td>4xx</td>
<td>All 4xx status codes, except 403, 404, 410</td>
</tr>
<tr>
<td>5xx</td>
<td>All 5xx status codes</td>
</tr>
</tbody>
</table>
### 304-Hits

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>304-Hit</td>
<td>If a cached item is unchanged since the last time it was requested, the NetScaler serves a 304 Not Modified response. The 304 Not Modified Header fields inserted are Date, Entity Tag (ETag), Expires, Cache-Control and/or Vary.</td>
</tr>
<tr>
<td>RFC 2616 section 13.3</td>
<td>When a cache has a stale entry that it would like to use as a response to a client’s request, it first has to check with the origin server (or possibly an intermediate cache with a fresh response) to see if its cached entry is still usable. We call this “validating” the cache entry. Since we do not want to have to pay the overhead of retransmitting the full response if the cached entry is good, and we do not want to pay the overhead of an extra round trip if the cached entry is invalid, the HTTP/1.1 protocol supports the use of conditional methods. The key protocol features for supporting conditional methods are those called “cache validators.” When an origin server generates a full response, it attaches some sort of validator to it, which is kept with the cache entry. When a client (user agent or proxy cache) makes a conditional request for a resource for which it has a cache entry, it includes the associated validator in the request. The server then checks that validator against the current validator for the entity, and, if they match (see section 13.3.3 of the RFC), it responds with a special status code (usually a 304 Not Modified) and no entity-body. Otherwise, it returns a full response (including entity-body). Thus, we avoid transmitting the full response if the validator matches, and we avoid an extra round trip if it does not match. In HTTP/1.1, a conditional request looks exactly the same as a normal request for the same resource, except that it carries a special header (which includes the validator) that implicitly turns the method (usually, GET) into a conditional. A “use” of a validator is either when a client generates a request and includes the validator in a validating header field, or when a server (or NetScaler) compares two validators.</td>
</tr>
<tr>
<td>Term</td>
<td>Affected By</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Server Specified Expiration</td>
<td>Origin Server</td>
</tr>
<tr>
<td>Heuristic Aging</td>
<td>NetScaler Content Group</td>
</tr>
<tr>
<td>Absolute or Relative</td>
<td>NetScaler Content Group</td>
</tr>
<tr>
<td>Age</td>
<td>NetScaler Cache</td>
</tr>
<tr>
<td>Via</td>
<td>NetScaler Cache</td>
</tr>
<tr>
<td>Cache-Control</td>
<td>NetScaler Cache</td>
</tr>
<tr>
<td>ETag (Entity-Tag)</td>
<td>NetScaler Cache</td>
</tr>
</tbody>
</table>
Application Profiling

Taking a trace

Running a trace will help you ‘profile’ the application. It is recommended that you do this before placing the Citrix Application Switch in-line of the Application traffic. This will gather important information about the Application that will help you understand its basic operation at Layer 7, and help you begin to understand what it is that needs to be cached.

Running a trace exposes the flow of transactions between all points of interest. Traces are especially helpful when digging in to find what is contained within the headers being exchanged between the client and the application.

For Requests from the client, look at the GET Header for cacheable objects, or objects that will likely be cache ‘hits’. In other words, objects that might be well served from cache.

For Responses from the server, look at the Content-Type: Header for cacheable objects.

Taking a trace with wireshark

The free network protocol analyzer called wireshark, http://www.wireshark.org, will capture packets for you on the localhost, whether it's windows or linux. By filtering the stream of packets by IP Address, right clicking and selecting ‘Follow TCP Stream’ inside of wireshark, you can see the headers for both requests and responses.
Taking a trace with the Citrix Application Switch

Running a trace will expose the flow of transactions between all points of interest, especially the client, load balancing VIPs and backend servers. Traces are especially helpful when digging in to find out if the proper headers are being exchanged between client <-> VIP and VIP <-> backend servers. A trace can be run directly on the Citrix Application Switch. Once downloaded this file can be opened and request and response headers read with Wireshark, a free network trace utility, http://www.wireshark.org.

Viewing headers with Paros

Paros was originally written for web security, but has value when viewing request and response headers, cookies and the like. Through Paros’s proxy nature, all HTTP and HTTPS data between server and client, including cookies and form fields, can be intercepted. There is an additional option of trapping and modifying data before sending it on to the server, or client. Paros can be found at http://parosproxy.org.

Viewing headers with Live HTTP Headers

Live HTTP Headers, http://livehttpheaders.mozdev.org/, was developed for use with the Firefox web browser. It is a free add-on and allows you to view HTTP header information in real time.

Viewing headers with IE Analyzer

IEInspector HTTP Analyzer, http://www.ieinspector.com, is a tool that allows you to monitor, trace, debug and analyze HTTP/HTTPS traffic in real-time. It works with Microsoft Internet Explorer.

Viewing headers with IE Watch

IEWatch, http://www.iewatch.com, is another plug-in for Microsoft Internet Explorer that helps you profile your web applications. You can use this tool to dig deep into the inner workings of web applications to find hidden issues.
Caching - Static

Static Caching

All applications are different, which is why we taught you how to profile an application in the preceding section. The content that needs to be cached for one application might be different for others. By running a trace and profiling the application, you can get an idea of what content to cache on requests and responses. You also get an idea of what content could be served from cache, when client requests come into the Application Switch.

There are two types of caching on the Application Switch, static and dynamic. You can add your own static cache policies and can create separate content groups. The difference between a static cache content group and a dynamic cache content group is the dynamic cache content group uses parameters. A DEFAULT content group already exists and by enabling caching, many static objects will get cached automatically.

One way to determine if the backend Application is sending content that is cacheable, is to observe the Cache-Control header in the responses it sends. If static content is considered cache-able, an HTTP response header “Cache-Control:” will have a ‘max-age’ parameter sent along with the object. If it is not intended to be cached, it will have a ‘no-cache’ or ‘private’ value in this field.

The quickest way to cache static content plus what the Application is intending to be cached is to 1) turn on the default caching rules and 2) create policies that cache anything the application intended to cache. Most applications will follow the rules laid out in RFC 2616 for caching, section 13 & section 14.

What is not cacheable

A response is not cacheable if the request method, request header fields or the response header fields indicate it is non-cacheable. If the request is a POST, PUT or DELETE method, the response is not cacheable. In fact, POST, PUT and DELETE should be used as invalidation directives. If the request header contains a Cache-Control directive to not-cache, the response is not cacheable. If the response contains a Cache-Control directive to not-cache, the response is not cacheable. There are cases where you don’t want to cache Authorization or Cookie information either.

In the case of the Oracle Enterprise Business Suite 12 Application, it uses a Java based forms application. When this forms application is launched you will see the User-Agent header change from IE or Firefox to Java, and will most likely have along with it a request header Cache-Control: no-cache. When this happens, the EBS12 client is basically telling all caches in the path that it must talk directly to the origin and the server must not use a cached copy when responding to the request. Another client request, Cache-Control: max-age=0 forces each cache along the path to the origin server to revalidate its own entry with the next cache or origin server. The validation can be a conditional request.

In addition to the Citrix built-in no-cache policies, you should pay attention to what the backend application is marking as non-cacheable. Applications use the Cache-Control header to insert directives to cache’s along the path of the content, on it’s way to the client. If this header says no-cache, no-store, or private intended only for that user, don’t cache it. These should be programmed in as custom no-cache policies.

Certain Hosts, URLs or Parameters you may not want to be cached. These need to be added as custom no-cache policies and should be marked to Precede the Default cache policies.

NOTE:

The default integrated caching policies are not visible to the end-user in v8.0, so there is no way to tell if they are working. You can create your own policies and override the defaults, then you will be able to see the hit counters. In v9.0+, the default integrated caching policies are visible, they all start with the underscore character, for example “_nonGetRequest”.
### Citrix built-in no-cache policies

<table>
<thead>
<tr>
<th>Flow</th>
<th>Header</th>
<th>Contains</th>
<th>Expression</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>Method</td>
<td>non-GET</td>
<td>METHOD != GET</td>
<td>NOCACHE</td>
</tr>
<tr>
<td>request</td>
<td>Authorization</td>
<td></td>
<td>REQ.HTTP:HEADER Authorization EXISTS</td>
<td>NOCACHE</td>
</tr>
<tr>
<td>request</td>
<td>Proxy-Authorization</td>
<td></td>
<td>REQ.HTTP:HEADER Proxy-Authorization EXISTS</td>
<td>NOCACHE</td>
</tr>
<tr>
<td>request</td>
<td>Cookie</td>
<td></td>
<td>HEADER Cookie EXISTS</td>
<td>NOCACHE</td>
</tr>
<tr>
<td>response</td>
<td>Pragma</td>
<td>contains</td>
<td>HEADER Pragma CONTAINS no-cache</td>
<td>NOCACHE</td>
</tr>
<tr>
<td>req/resp</td>
<td>Cache-Control</td>
<td>no-cache</td>
<td>HEADER Cache-Control CONTAINS no-cache</td>
<td>NOCACHE</td>
</tr>
<tr>
<td>req/resp</td>
<td>Cache-Control</td>
<td>nostore</td>
<td>HEADER Cache-Control CONTAINS nostore</td>
<td>NOCACHE</td>
</tr>
<tr>
<td>req/resp</td>
<td>Cache-Control</td>
<td>private</td>
<td>HEADER Cache-Control CONTAINS private</td>
<td>NOCACHE</td>
</tr>
</tbody>
</table>

### Custom no-cache policies

<table>
<thead>
<tr>
<th>Flow</th>
<th>Header</th>
<th>Contains</th>
<th>Expression</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>Set Cookie</td>
<td></td>
<td>RES.HTTP:HEADER Set Cookie EXISTS</td>
<td>NOCACHE</td>
</tr>
<tr>
<td>response</td>
<td>Set Cookie2</td>
<td></td>
<td>RES.HTTP:HEADER Set Cookie2 EXISTS</td>
<td>NOCACHE</td>
</tr>
<tr>
<td>request</td>
<td>Host</td>
<td>accel91</td>
<td>REQ.HTTP:HEADER Host CONTAINS accel91.citrix.com</td>
<td>NOCACHE</td>
</tr>
</tbody>
</table>

To add custom NOCACHE policies, from the GUI, navigate to NetScaler ➡️ Integrated Caching ➡️ Policies ➡️ Add.

Click on Activate Policies and Add the new policy to the Configured list. Give it a priority, and Custom No-Cache policies should Precede Default Rules.
What is cacheable

By default, a response is cacheable if the request method, request header fields and the response header fields indicate it is cacheable. As long as the request is a GET method and doesn’t contain any Cache-Control directives to not cache anything, the response can be cached. Default cache policies place responses in the DEFAULT cache group.

Another set of custom policies that are important, are ones that look at the responses coming from the backend application that it deems appropriate to cache. If the application says it’s cacheable, then we should take the liberty to cache it also.

The default integrated caching policies will look for images such as .gif, .jpeg, .png and cache them; however, there are other types of content that are regarded by some users as cacheable. For example, .css, .js and other document file types that tend to remain static. These can be added as custom policies.

NOTE:
The DEFAULT cache content group satisfies the majority of caching requirements. If you feel it necessary to adjust prefetch timers, freshness timers and the like you can create additional content groups, and have responses placed into them.

Citrix built-in cache policies - placed into DEFAULT group

<table>
<thead>
<tr>
<th>Flow</th>
<th>Header</th>
<th>Contains</th>
<th>Expression</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>Content-Type</td>
<td>image</td>
<td>RES.HTTP.HEADER Content-Type CONTAINS image</td>
<td>CACHE</td>
</tr>
<tr>
<td>response</td>
<td>Expires</td>
<td></td>
<td>RES.HTTP.HEADER Expires EXISTS</td>
<td>CACHE</td>
</tr>
<tr>
<td>response</td>
<td>Cache-Control</td>
<td>public</td>
<td>RES.HTTP.HEADER Cache-Control CONTAINS public</td>
<td>CACHE</td>
</tr>
<tr>
<td>response</td>
<td>Cache-Control</td>
<td>max-age</td>
<td>RES.HTTP.HEADER Cache-Control CONTAINS max-age</td>
<td>CACHE</td>
</tr>
<tr>
<td>response</td>
<td>Cache-Control</td>
<td>s-maxage</td>
<td>RES.HTTP.HEADER Cache-Control CONTAINS s-maxage</td>
<td>CACHE</td>
</tr>
<tr>
<td>response</td>
<td>Cache-Control</td>
<td>must-revalidate</td>
<td>RES.HTTP.HEADER Cache-Control CONTAINS must-revalidate</td>
<td>CACHE</td>
</tr>
<tr>
<td>response</td>
<td>Cache-Control</td>
<td>proxy-revalidate</td>
<td>RES.HTTP.HEADER Cache-Control CONTAINS proxy-revalidate</td>
<td>CACHE</td>
</tr>
</tbody>
</table>

Custom cache policies - common cache policies

<table>
<thead>
<tr>
<th>Flow</th>
<th>Header</th>
<th>Contains</th>
<th>Expression</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>Content-Type</td>
<td>css</td>
<td>RES.HTTP.HEADER Content-Type CONTAINS css</td>
<td>CACHE</td>
</tr>
<tr>
<td>response</td>
<td>Content-Type</td>
<td>javascript</td>
<td>RES.HTTP.HEADER Content-Type CONTAINS javascript</td>
<td>CACHE</td>
</tr>
<tr>
<td>response</td>
<td>Content-Type</td>
<td>word</td>
<td>RES.HTTP.HEADER Content-Type CONTAINS word</td>
<td>CACHE</td>
</tr>
<tr>
<td>response</td>
<td>Content-Type</td>
<td>excel</td>
<td>RES.HTTP.HEADER Content-Type CONTAINS excel</td>
<td>CACHE</td>
</tr>
<tr>
<td>response</td>
<td>Content-Type</td>
<td>powerpoint</td>
<td>RES.HTTP.HEADER Content-Type CONTAINS powerpoint</td>
<td>CACHE</td>
</tr>
<tr>
<td>response</td>
<td>Content-Type</td>
<td>pdf</td>
<td>RES.HTTP.HEADER Content-Type CONTAINS pdf</td>
<td>CACHE</td>
</tr>
</tbody>
</table>
To add custom CACHE policies, from the GUI, navigate to NetScaler ➔ Integrated Caching ➔ Policies ➔ Add.

Click on Activate Policies and Add the new policy to the Configured list. Give it a priority, and Custom Cache policies should Precede Default Rules.
Caching - Dynamic

Dynamic Caching

The Citrix Application Switch can perform Dynamic content caching. Dynamic content is typically marked as non-cacheable by web servers and applications because it always changing. For example, an inventory report, or a general ledger report that is run periodically and viewed by one or more employees. Dynamic content can also be classified as content that is only viewable by a select few users or made available to specific users during their ‘session’.

In order to perform Dynamic caching, the Citrix Application Switch needs to key off of certain fields in the headers that are exchanged between the client and server. Once again, by running a trace and profiling the application, you can get an idea of not only what content to cache but more importantly what header information to build dynamic caching policies with.

The difference between a static cache content group and a dynamic cache content group is the dynamic cache content group uses parameters. You can create a new content group for dynamic responses or place them in the DEFAULT group.

Dynamic Cache Selectors

A best practice for building dynamic caching policies is to use 3 or 4 header fields to key off of, in the responses from the server. The fields you use to match on, in the headers of the requests and responses are called “Selectors”, and these are configurable in the Citrix Application Switch. Selectors are used to “HIT” or match cache entries which are categorized according to the selectors you have chosen. Selectors are also used to ‘INVALIDATE’ or expire the cache, outside of the timer based expiration methods. The Integrated Cache will automatically cache based on Hostname and IP address, but you need to specify a few more fields for dynamic caching. You can use any number and type of header fields to create selectors, but these are a good start:

- URL
- Request Parameter(s)
- Cookie Value(s)

Referring to our reference application, Oracle Enterprise Business Suite 12, we take a trace and login to record the events and can see that there are some good values to use for dynamic caching.

NOTE:

The trace in this example was taken using Wireshark ‘Follow TCP Stream’. Client requests are in Red, Server responses are in Blue.

GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.2; .NET CLR 1.1.4322; .NET CLR 2.0.50727)
Host: accel82.citrix.com:8000
Connection: Keep-Alive
Cookie: oracle-uix=0;^GMT-7:00^p

HTTP/1.1 200 OK
Date: Fri, 18 Apr 2008 23:08:42 GMT
Server: Oracle-Application-Server-10g/10.1.3.0.0 Oracle-HTTP-Server
Last-Modified: Sat, 05 Apr 2008 21:21:11 GMT
ETag: "2c4644-516-47f7ed47" Accept-Ranges: bytes
Content-Length: 1302
Keep-Alive: timeout=15
Connection: Keep-Alive
Content-Type: text/html

GET /OA_HTML/AppsLogin HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.2; .NET CLR 4.0.21977; .NET CLR 2.0.50727)
Host: accel82.citrix.com:8000
Connection: Keep-Alive
Cookie: oracle-uix=0;^GMT-7:00^p

HTTP/1.1 302 Moved Temporarily
Date: Fri, 18 Apr 2008 23:08:43 GMT
Server: Oracle-Application-Server-10g/10.1.3.0.0 Oracle-HTTP-Server
Cache-Control: no-cache
Pragma: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Location: http://accel82.citrix.com:8000/OA_HTML/AppsLocalLogin.jsp
Keep-Alive: timeout=15
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/plain

HTTP/1.1 302 Moved Temporarily
Date: Fri, 18 Apr 2008 23:08:43 GMT
Server: Oracle-Application-Server-10g/10.1.3.0.0 Oracle-HTTP-Server
Set-Cookie: CITRIX=uANsV6jhUFsrKDm8AR; Domain=.*.citrix.com; Path=/
Location: http://accel82.citrix.com:8000/OA_HTML/RF.jsp?function_id=1032924&resp_id=-1&resp_appl_id=-1&security_group_id=0&lang_code=US&params=O3aPQplgodi5k-2XsDnw7w&oas=fZ9hHAQsDrn-W8VTekmCoxHg..
Keep-Alive: timeout=15
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

GET /OA_HTML/RF.jsp?function_id=1032924&resp_id=-1&resp_appl_id=-1&security_group_id=0&lang_code=US&params=O3aPQplgodi5k-2XsDnw7w&oas=fZ9hHAQsDrn-W8VTekmCoxHg.. HTTP/1.1
Accept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg, */*
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.2; .NET CLR 1.1.4322; .NET CLR 2.0.50727)
Host: accel82.citrix.com:8000
Connection: Keep-Alive
Cookie: oracle.uix=0^GMT-7:00^p

HTTP/1.1 200 OK Date: Fri, 18 Apr 2008 23:08:43 GMT
Server: Oracle-Application-Server-10g/10.1.3.0.0 Oracle-HTTP-Server
Content-Location: http://accel82.citrix.com:8000/OA_HTML/OA.jsp
Set-Cookie: JSESSIONID=4361fd5253fcee562e714fe1411928688f618ec644.s3400394.heheN0Q0Pa0x00Kbzh0; path=/OA_HTML
Keep-Alive: timeout=15
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

Initial Oracle login screen after redirect has been issued.
Oracle Navigator

From the trace, just by logging into the Oracle Application, we already have some valuable information from which we can build selectors to be used in our dynamic caching rules. The application issued a couple of Set-Cookie commands to the client, telling the client these are the unique session id’s. From this trace, we can determine that we need the following HIT selectors:

Oracle Navigator selectors - Hit group 1

<table>
<thead>
<tr>
<th>Selector</th>
<th>Header Field</th>
<th>Contains</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>OANavHit1</td>
<td>Method</td>
<td>GET</td>
<td>HTTP.REQ.METHOD.EQ(&quot;GET&quot;)</td>
</tr>
<tr>
<td>OANavHit1</td>
<td>URL</td>
<td>RF.jsp</td>
<td>HTTP.REQ.URL.CONTAINS(&quot;/RF.jsp&quot;)</td>
</tr>
<tr>
<td>OANavHit1</td>
<td>Parameter</td>
<td>function_id</td>
<td>HTTP.REQ.URL.QUERY.VALUE(&quot;function_id&quot;)</td>
</tr>
<tr>
<td>OANavHit1</td>
<td>Cookie</td>
<td>CITRIX</td>
<td>HTTP.REQ.COOKIE.VALUE(&quot;CITRIX&quot;)</td>
</tr>
<tr>
<td>OANavHit1</td>
<td>Cookie</td>
<td>JSESSIONID</td>
<td>HTTP.REQ.COOKIE.VALUE(&quot;JSESSIONID&quot;)</td>
</tr>
</tbody>
</table>

If we continue tracing the Oracle Application and click through more menus we see that there are some additional header fields that can be used as Hit selectors. The same Invalidation selectors can be used.

Oracle Navigator selectors - Hit group 2

<table>
<thead>
<tr>
<th>Selector</th>
<th>Header Field</th>
<th>Contains</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>OANavHit2</td>
<td>Method</td>
<td>GET</td>
<td>HTTP.REQ.METHOD.EQ(&quot;GET&quot;)</td>
</tr>
<tr>
<td>OANavHit2</td>
<td>URL</td>
<td>/OA.jsp</td>
<td>HTTP.REQ.URL.CONTAINS(&quot;/OA.jsp&quot;)</td>
</tr>
<tr>
<td>OANavHit2</td>
<td>Parameter</td>
<td>OAFunc</td>
<td>HTTP.REQ.URL.QUERY.VALUE(&quot;OAFunc&quot;)</td>
</tr>
<tr>
<td>OANavHit2</td>
<td>Cookie</td>
<td>CITRIX</td>
<td>HTTP.REQ.COOKIE.VALUE(&quot;CITRIX&quot;)</td>
</tr>
<tr>
<td>OANavHit2</td>
<td>Cookie</td>
<td>JSESSIONID</td>
<td>HTTP.REQ.COOKIE.VALUE(&quot;JSESSIONID&quot;)</td>
</tr>
</tbody>
</table>

After determining the Hit selectors, it is appropriate to determine what the invalidation selectors should be. In other words, what header fields should we key off of to invalidate or flush the entries in the cache. This most certainly happens on a Post method as the client is sending data into the application to be changed. You don’t need the same number of invalidation selectors as you have for hit selectors.

Oracle Navigator selectors - Invalidation group

<table>
<thead>
<tr>
<th>Selector</th>
<th>Header Field</th>
<th>Contains</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>OANavInv1</td>
<td>Method</td>
<td>POST</td>
<td>HTTP.REQ.METHOD.EQ(&quot;POST&quot;)</td>
</tr>
<tr>
<td>OANavInv1</td>
<td>Cookie</td>
<td>CITRIX</td>
<td>HTTP.REQ.COOKIE.VALUE(&quot;CITRIX&quot;)</td>
</tr>
</tbody>
</table>
GET /OA_HTML/OA.jsp?OAFunc=OAHOMEPAGE HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/x-shockwave-flash, application/xaml+xml, application/vnd.msexcel, application/vnd.ms-powerpoint, application/msword, application/x-silverlight, */*
Referer: http://accel82.citrix.com:8000/OA_HTML/RF.jsp?function_id=1032924&resp_id=-1&resp_appl_id=-1&security_group_id=0&lang_code=US&params=O3aPQplgodi5k-2XsDnw7w&oa=Ngu1wehOX_iYOwbtJnhnHbw..
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30; .NET CLR 1.1.4322; .NET CLR 3.0.04506.648; InfoPath.2)
Host: accel82.citrix.com:8000
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: JSESSIONID=4361fd5253fc5412307d2b74d3884bff982afe803ce.e34Oc30aheNb00Paxa0b3b0; oracle.uidx=0^GMT-7:00^p; CITRIX=ZJQrywjbT3imkA6GnosRm-Y6:S
HTTP/1.1 200 OK
Date: Wed, 09 Apr 2008 21:25:40 GMT
Server: Oracle-Application-Server-10g/10.1.3.0.0 Oracle-HTTP-Server
Content-Location: http://accel82.citrix.com:8000/OA_HTML/OA.jsp
Set-Cookie: JSESSIONID=4361fd5253fc5412307d2b74d3884bff982afe803ce.e34Oc30aheNb00Paxa0b3b0; path=/OA_HTML
Keep-Alive: timeout=15
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

OA.jsp links - clicking these displays the RF.jsp links

RF.jsp links - clicking these launches the Java forms.
To add Dynamic caching Hit selector for the RF menus, from the GUI, navigate to NetScaler ➔ Integrated Caching ➔ Cache Selectors ➔ Add. Add each expression based on the table above. Click on Create.

Create a Dynamic caching Invalidation selector for the RF and OA Menus. In this example, OANavInv1.

To engage the Dynamic caching mechanism, you need to create a content Group that uses the selectors. Navigate to NetScaler ➔ Integrated Caching ➔ Content Groups ➔ Add.

Add a new content group. Check the radio button at the bottom of the ‘Selection’ tab for Selector-based.

From the drop down menu select the OANavHit1 selector for Cache Hits, and the OANavInv1 selector for Cache Invalidation.
To add Dynamic caching Hit selector for the OA menus, from the GUI, navigate to NetScaler ➤ Integrated Caching ➤ Cache Selectors ➤ Add. Add each expression based on the table above. Click on Create.

We can use the same Invalidation selector, OANavInv1.

Create another Dynamic content Group for the OA Menus. Navigate to NetScaler ➤ Integrated Caching ➤ Content Groups ➤ Add.

Add a new content group. Check the radio button at the bottom of the ‘Selection’ tab for Selector-based.

From the drop down menu select the OANavHit2 selector for Cache Hits, and the OANavInv1 selector for Cache Invalidation.
Oracle Forms

Oracle runs a large part of their application over a Java based forms program. The forms are launched with GET methods. The Input to the Oracle Application is done through POST methods. Dynamic cache Hit selectors can be built on the GET method and Invalidation selectors can be built on the POST method.
Oracle Forms selectors - Hit group

<table>
<thead>
<tr>
<th>Selector</th>
<th>Header Field</th>
<th>Contains</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>OAFormsHit</td>
<td>Method</td>
<td>GET</td>
<td>HTTP.REQ.METHOD.EQ(&quot;GET&quot;)</td>
</tr>
<tr>
<td>OAFormsHit</td>
<td>Parameter</td>
<td>/forms/lservlet</td>
<td>HTTP.REQ.URL.CONTAINS(&quot;/forms/lservlet&quot;)</td>
</tr>
<tr>
<td>OAFormsHit</td>
<td>Cookie</td>
<td>CITRIX</td>
<td>HTTP.REQ.COOKIE.VALUE(&quot;CITRIX&quot;)</td>
</tr>
</tbody>
</table>

Oracle Forms selectors - Invalidation group

<table>
<thead>
<tr>
<th>Selector</th>
<th>Header Field</th>
<th>Contains</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>OAFormsInv</td>
<td>Method</td>
<td>POST</td>
<td>HTTP.REQ.METHOD.EQ(&quot;POST&quot;)</td>
</tr>
<tr>
<td>OAFormsInv</td>
<td>Parameter</td>
<td>/forms/lservlet</td>
<td>HTTP.REQ.URL.CONTAINS(&quot;/forms/lservlet&quot;)</td>
</tr>
<tr>
<td>OAFormsInv</td>
<td>Cookie</td>
<td>CITRIX</td>
<td>HTTP.REQ.COOKIE.VALUE(&quot;CITRIX&quot;)</td>
</tr>
</tbody>
</table>

NOTE:

Forms in Oracle E-Business Suite are supported in Internet Browsers and launched through a separate browser window frame. Therefore, even with tabs enabled, the forms navigator and the forms windows will continue to open in a new window and not in a new tab. This is intended functionality on the part of the Oracle application. Be on the lookout for Pop-Up messages and Install Active-X client messages in the browser window. For more information refer to Oracle Metalink Document ID: 285218.1.

Add Dynamic caching Hit selector & Invalidation selector for the Oracle forms.

Create another Dynamic content Group for the Oracle Forms.

Use the selectors OAFormsHit for Cache Hits, and the OAFormsInv for Cache Invalidation.
Oracle Reporting

Oracle runs a large part of their application over a Java based forms program. Much of the input and output to Oracle is performed through the forms program. Many of the reports that are generated are requested from within this forms program. Whether the report is output in text or in pdf format, it is sent to the client through the forms program over HTTP using port 8000. Trying to find some fields to key off of for dynamic caching is a challenge, because the reporting system in Oracle uses a query parameter called ‘temp_id’ as input to an executable in the request.

The ‘temp_id’ parameter is used in the URL that gets generated for the call to the Web Report Review agent (FNDWRR). The ‘temp_id’ string is part of an encryption mechanism that does not recognize or record the first time the request is accessed and then regenerates the temp_id string in order to be used again. According to Oracle, this URL will only work one time, as this is a security feature. The temp_id is stored in a table, and once it is used, it is deleted.

This makes the attempt of caching Oracle reports in the Citrix Applicaiton Switch difficult to track as the temp_id is always changing, even for the same report. As you can see from the response in Blue, the actual report filename is sent with the file, so it is conceivable that a server process could be written to copy the file to a separate directory, and be downloaded using the filename instead of the temp_id parameter. This way, upon accessing the report several times, with the same filename, would take advantage of dynamic caching.

GET /OA_CGI/FNDWRR.exe?temp_id=3236120438 HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30; .NET CLR 1.1.4322; .NET CLR 3.0.04506.648; InfoPath.2)
Host: accel82.citrix.com:8000
Connection: Keep-Alive
Cookie: oracle.uix=0^^GMT-7:00^p; CITRIX=oQ5I-R-3VouAtXw8MxuLRVSx:S

HTTP/1.1 200 OK
Date: Wed, 09 Apr 2008 21:45:10 GMT
Server: Oracle-Application-Server-10g/10.1.3.0.0 Oracle-HTTP-Server
Content-disposition: inline; filename ="Account_Analysis_132_Char_090408.pdf"
FNDFS-Error: 0
Content-length: 14276513
Keep-Alive: timeout=15
Connection: Keep-Alive
Content-Type: application/pdf

Report request from ‘View Output’. Report is accessed via ‘temp_id’ parameter in the GET request. However, it is sent in the response with the full filename ‘Account_Analysis_132_Char_090408.pdf’. If you save the report, is will save as ‘FNDWRR.pdf’.
Cracking the Oracle Reporting Nut:

As the preceding section mentions, Oracle uses a randomly generated “temp_id” parameter to access the reports on the Application server. However, according to Oracle Metalink Document 28130.1, the temp_id parameter matches the file_id that the file is temporarily stored in.

Some other interesting deductions can be formed. If a user clicks on the ‘View Log’ for a specific report, Oracle will send back a log file of the report generation, and in that text file report is the ‘Request ID’ appertaining to that report.

It just so happens that reports in .txt file format are stored on the Oracle Application server in the directory ‘/u01/applmgr/CITRIX/inst/apps/CITRIX_accel82/logs/ora/10.1.2/reports/cache’ directory under the filename ‘o<ConcReqId>xxxxxxx.txt’, where ConcReqId is the Request Id used to generate the report (the same value from the View Log output).

Reports in .pdf file format are stored on the Oracle Application server in the directory ‘/u01/applmgr/CITRIX/inst/apps/CITRIX_accel82/logs/appl/conc/out’ under the filename ‘<ModuleName>_<ConcReqid>_x.pdf’, where ConcReqId is the Request id used to generate the report (the same value from the View Log output).

Another approach is to copy the .txt and .pdf files to the $OA_HTML or ‘/u01/applmgr/CITRIX/apps/apps_st/comn/webapps/oacore/html’ directory and access them directory from a browser using the Request Id, for example,

On the Oracle Server:
- cd /u01/applmgr/CITRIX/inst/apps/CITRIX_accel82/logs/appl/conc/out
- cp GLACTANL_4446138_1.pdf $OA_HTML

From the Client Browser:

(where 4446138 is the Request Id used to generate the report)

Specialized Report Cache Headers:

In the proof of concept engagement we did for this deployment guide, we had the end-user initiate requests to View Reports with an added Request Header, so that our Dynamic Caching Hit Selector could key off of it. The custom header was called "REPORT: xxxxxxx", where xxxxxxx was a unique session & report identifier. The result being that once the Oracle report was downloaded and viewed one time, it was cached, and by using the Hit selector for the Header "REPORT: xxxxxxx", it could be served from cache, reducing the time to generate the report and the need to access the backend server.
Oracle Logout Invalidation

Content doesn’t remain fresh forever. Some clients stay logged in for hours, some only for a few minutes. One thing for sure, is that if a client logs out, the cache should be emptied. If the client selects the Logout link in Oracle, invalidate the cache.

GET /OA_HTML/OALogout.jsp?menu=Y HTTP/1.1
Accept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg, */*
Referer: http://accel82.citrix.com:8000/OA_HTML/OA.jsp?OAFunc=OAHOMEPAGE
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.2; .NET CLR 1.1.4322; .NET CLR 2.0.50727)
Host: accel82.citrix.com:8000
Connection: Keep-Alive
Cookie: JSESSIONID=4361fd5253fc4857039fbb4242249f0b52d747c8571.e340c3000aheNhQ00PaxaObx0Kbxb0; oracle.uix=0^GMT-7:00^p; CITRIX=FaFckTfkWXFu2PZq2H9SM73:S

HTTP/1.1 302 Moved Temporarily
Date: Mon, 21 Apr 2008 23:18:00 GMT
Server: Oracle-Application-Server-10g/10.1.3.0.0 Oracle-HTTP-Server
Set-Cookie: JSESSIONID=4361fd5253fc4857039fbb4242249f0b52d747c8571.e340c3000aheNhQ00PaxaObx0Kbxb0; path=/OA_HTML
Location: http://accel82.citrix.com:8000/OA_HTML/AppsLogout
Keep-Alive: timeout=15
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
To add a Logout Invalidation selector, from the GUI, navigate to NetScaler ➔ Integrated Caching ➔ Cache Selectors ➔ Add. Add each expression based on the table above. Click on Create.

An “Invalidation” policy must be created to use the logout selector. From the GUI, navigate to NetScaler ➔ Integrated Caching ➔ Policies ➔ Add. Enter in an expression that matches on the URL /OALogout.jsp

Under the Invalidation Parameters tab, Invalidate selected objects in the following parameterized groups using the ‘OALogout’ Selector.

Activate the Invalidation Policy and it needs to Precede Default Rules.
Appendix A - NetScaler Application Switch Configuration

Primary NetScaler

> #NS8.0 Build 53.2
# Last modified by 'save config', Tue Apr 29 21:41:56 2008
set ns config -IPAddress 10.217.104.71 -netmask 255.255.255.0
enable ns feature WL SP IC
enable ns mode FR L3 Edge USNIP PMTUD
set cache parameter -memLimit 512 -via "NS-CACHE-8.0:   1" -verifyUsing HOSTNAME_AND_IP -maxPostLen 0 -prefetchMaxPending 4294967294 -enableBypass YES
add cache selector OANavHit1 "HTTP.REQ.METHOD.EQ("GET")" "HTTP.REQ.URL.CONTAINS("/RF.jsp")" "HTTP.REQ.URL.QUERY.VALUE("function_id")" "HTTP.REQ.COOKIE.VALUE("CITRIX")" "HTTP.REQ.COOKIE.VALUE("JSESSIONID")"
add cache selector OANavHit2 "HTTP.REQ.METHOD.EQ("GET")" "HTTP.REQ.COOKIE.VALUE("CITRIX")" "HTTP.REQ.COOKIE.VALUE("JSESSIONID")" "HTTP.REQ.URL.CONTAINS("/OA.jsp")" "HTTP.REQ.URL.QUERY.VALUE("OAFunc")"
add cache selector OANavInv1 "HTTP.REQ.METHOD.EQ("POST")" "HTTP.REQ.COOKIE.VALUE("CITRIX")"
add cache selector OAFormsHit "HTTP.REQ.METHOD.EQ("GET")" "HTTP.REQ.URL.CONTAINS("/forms/lservlet")" "HTTP.REQ.COOKIE.VALUE("CITRIX")"
add cache selector OAFormsInval "HTTP.REQ.URL.CONTAINS("/forms/lservlet")" "HTTP.REQ.COOKIE.VALUE("CITRIX")" "HTTP.REQ.METHOD.EQ("POST")"
add cache selector OALogout "HTTP.REQ.METHOD.EQ("GET")" "HTTP.REQ.URL.CONTAINS("/OALogout.jsp")" "HTTP.REQ.COOKIE.VALUE("CITRIX")"
set cache contentGroup BASEFILE -relExpiry 86000 -maxResSize 256 -memLimit 2
set cache contentGroup DELTAJS -relExpiry 86000 -insertAge NO -maxResSize 256 -memLimit 1 -pinned YES
add cache contentGroup DynamicOA1 -heurExpiryParam 10 -removeCookies NO -prefetchMaxPending 0 -ignoreReqCachingHdrs NO -hitSelector OANavHit1 -invalSelector OANavInv1
add cache contentGroup DynamicOA2 -heurExpiryParam 10 -removeCookies NO -prefetchMaxPending 0 -ignoreReqCachingHdrs NO -hitSelector OANavHit2 -invalSelector OANavInv1
add cache contentGroup DynamicOAForms -heurExpiryParam 10 -removeCookies NO -prefetchMaxPending 0 -ignoreReqCachingHdrs NO -hitSelector OAFormsHit -invalSelector OAFormsInval
add cache policy OALogoutInvalidate -rule "REQ.HTTP.URL.CONTAINS("/OALogout.jsp")" -action INVAL -invalObjects DynamicOAForms,OALogout DynamicOA1,OALogout DynamicOA2,OALogout
add cache policy nocache_hosts -rule "REQ.HTTP.HEADER Host CONTAINS accel91.citrix.com || REQ.HTTP.HEADER Host CONTAINS accel88.citrix.com" -action NOCACHE
add cache policy req_ContentLength -rule "REQ.HTTP.HEADER Content-Length EXISTS" -action NOCACHE
add cache policy req_TE2 -rule "REQ.HTTP.HEADER Transfer-Encoding EXISTS" -action NOCACHE
add cache policy req_Auth -rule "REQ.HTTP.HEADER Authorization EXISTS" -action NOCACHE
add cache policy req_PAuth -rule "REQ.HTTP.HEADER Proxy-Authorization EXISTS" -action NOCACHE
add cache policy common_content -rule "RES.HTTP.HEADER Content-Type CONTAINS css || RES.HTTP.HEADER Content-Type CONTAINS javascript || RES.HTTP.HEADER Content-Type CONTAINS word || RES.HTTP.HEADER Content-Type CONTAINS excel || RES.HTTP.HEADER Content-Type CONTAINS powerpoint || RES.HTTP.HEADER Content-Type CONTAINS pdf" -action CACHE -storeInGroup DEFAULT
add cache policy resp_SetCookie -rule "RES.HTTP.HEADER Set-Cookie EXISTS || RES.HTTP.HEADER Set-Cookie2 EXISTS" -action NOCACHE
bind cache global OALogoutInvalidate -priority 20 -precedeDefRules YES
bind cache global nocache_hosts -priority 30 -precedeDefRules YES
bind cache global req_ContentLength -priority 10000
bind cache global req_TE2 -priority 10010
bind cache global req_Auth -priority 10020
bind cache global req_PAuth -priority 10030
bind cache global common_content -priority 19000 -precedeDefRules YES
bind cache global resp_SetCookie -priority 20000 -precedeDefRules YES
About Citrix

Citrix Systems, Inc. (Nasdaq:CTXS) is the global leader and the most trusted name in application delivery infrastructure. More than 200,000 organizations worldwide rely on Citrix to deliver any application to users anywhere with the best performance, highest security and lowest cost. Citrix customers include 100% of the Fortune 100 companies and 98% of the Fortune Global 500, as well as hundreds of thousands of small businesses and prosumers. Citrix has approximately 6,200 channel and alliance partners in more than 100 countries. Annual revenue in 2006 was $1.1 billion.

Citrix®, NetScaler®, GoToMyPC®, GoToMeeting®, GoToAssist®, Citrix Presentation Server™, Citrix Password Manager™, Citrix Access Gateway™, Citrix Access Essentials™, Citrix Access Suite™, Citrix SmoothRoaming™ and Citrix Subscription Advantage™ and are trademarks of Citrix Systems, Inc. and/or one or more of its subsidiaries, and may be registered in the U.S. Patent and Trademark Office and in other countries. UNIX® is a registered trademark of The Open Group in the U.S. and other countries.  Microsoft®, Windows® and Windows Server® are registered trademarks of Microsoft Corporation in the U.S. and/or other countries. All other trademarks and registered trademarks are property of their respective owners.

www.citrix.com