Improving the User Experience in SAP BI Platform 4.0 with Apache and WDeploy

Applies to:

SAP BI Platform 4.0. For more information, visit the Business Objects homepage.

Summary

This document proposes a framework for adding an Apache web server to your deployment of SAP BI Platform 4.0. By using Apache in conjunction with the WDeploy web application deployment tool, significant performance improvements become possible. These modifications reduce the server footprint of the web applications, and speed delivery of web content to client browsers.

Author: James Rapp

Company: SAP

Created on: 30 September 2011

Author Bio

James is a Senior Ecosystem Quality Manager at SAP working closely with OEM partners. James' experience with SAP BusinessObjects products stems from his previous experience as a Technical Account Manager for Large Enterprise deployments as well as an Onsite Technical Specialist in Business Objects Customer Assurance.

Table of Contents

Introduction

SAP BI Platform 4.0 provides a rich, AJAX enabled web application in the form of BI Launch Pad. From within BI Launch Pad, you can access Crystal reports, Web Intelligence documents, and other objects, and organize them to suit your preferences. It delivers a familiar "right-click" enabled interface for users to browse content and has a short ramp-up time in terms of user training.

This robust functionality does come at the cost of an increase of load to the application server due to the size and number of files. Some of this cost is offset by fairly fine grain cache-control that is now default in the application, and by 64-bit JVM which allow us to allocate more resources to Tomcat, or your application server of choice.

In this document, I will discuss how the addition of an Apache web server can improve end user interactions significantly and several issues that are difficult to resolve using a Java Application Server alone. Through the addition of WDeploy in split mode, which now successfully separates 99% of static content and deploys it to a web server, we can take advantage of some powerful Apache modules that accelerate and maximize the delivery of static content to shared web caches, such as proxies, and to clients not benefiting from browser cache (new users, users who have recently cleared their browser cache, VPN policies that clean cache upon access, etc).

Though this document was written for SAP BI Platform 4.0, the techniques described are also valid for use in SAP BusinessObjects Enterprise XI 3.1. However, in XI 3.1 they do not deliver the same degree of impact on performance. This is because the WDeploy application in XI 3.1 does not separate much of the static content that forms the backbone of InfoView. This static content is deployed by a java servlet out of JAR files in the InfoViewApp and InfoViewAppActions web applications.

Apache Web Server

Apache is the most popular web server in the world today running an estimated 63% of all web sites. It is an open-source HTTP Server for modern operating systems such as Linux, Unix, and Windows, and heavily extensible through a series of pre-compiled modules. Apache can be downloaded from the following location:

http://httpd.apache.org/download.cgi

The version of Apache used in this document is version 2.2.19.

WDeploy

The WDeploy tool is included with SAP BusinessObjects Business Intelligence platform to ease deployment of web applications to Java-based web application servers.

While each supported web application server requires different commands and web application package updates, WDeploy provides a consistent interface for administrators, and automates the adjustments needed for deployment to a specific web application server.

Documentation for WDeploy (4.0 SP02) can be found at:

http://service.sap.com/~sapidb/011000358700001237082010E/xi4_webappdeploy_win_en.pdf

Background

BI Launch Pad "out of the box" experience

SAP BI Platform 4.0 offers important enhancements to the default configuration of Tomcat and the bundled web applications. HTTP compression is now enabled on port 8080 by default. This is found by observing the addition of the following info in **bold** to the server.xml:

```
<Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443"
compression="on" URIEncoding="UTF-8" compressionMinSize="2048"
noCompressionUserAgents="gozilla, traviata"
compressableMimeType="text/html,text/xml,text/plain,text/css,text/javascript,text/json,ap
plication/json"/>
```

These properties direct Tomcat to compress all content, from any of the listed MIME types, larger than 2 KB before sending it to a client. With the processing power of most servers today, and widespread browser support for standard compression formats, HTTP compression should be enabled in almost every scenario. The additional CPU consumption required is generally negligible compared to the bandwidth saved, and the better user experience far outweighs the cost. Be sure to apply the same settings to your SSL connector if you're using one.

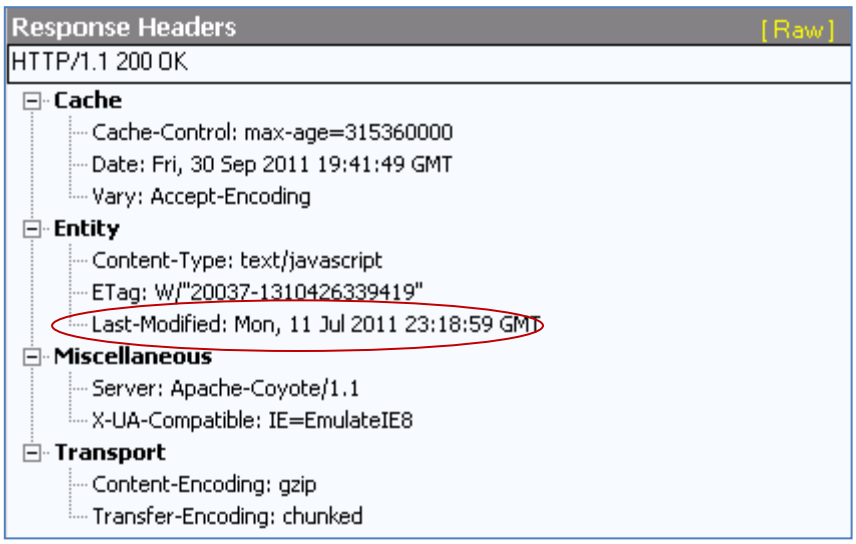
Another helpful enhancement is the addition of 'Cache-Control' headers and default 'max-age' values on the vast majority of content. This document assumes a basic understanding of Cache-Control but a very thorough tutorial can be found here for anyone new to the subject. In short, Cache-Control enables browsers, proxies, and other shared web caches to store copies of static content, such as CSS, JavaScript, and some images to improve performance and eliminate wasteful re-processing of requests by the application server.

In the XI 3.1 release, cache-control was possible, but required additional configuration of several web.xml files. The process is documented in SAP Note 1342390. With the advent of 4.0, the max-age parameter is set automatically to a value of 315360000, which is 10 years in seconds.

enterprise:8080	/BOE/portal/1107111813/InfoView/logon.jsp	2,402	
enterprise:8080	/BOE/portal/1107111813/InfoView/common/appService.do?service=skinning&resource=styles...	3,921	max-age=315360000
enterprise:8080	/BOE/portal/1107111813/InfoView/js/utils.js	5,475	max-age=315360000
enterprise:8080	/BOE/portal/1107111813/shared/js/caf/helpSystem.js	1,323	max-age=315360000
enterprise:8080	/BOE/portal/1107111813/InfoView/help/en/html/HelpFileMap.js	656	max-age=315360000

Note that the max-age parameter is configurable in 4.0 by modifying the property 'max.age' in the file
C:\Program Files (x86)\SAP BusinessObjects\Tomcat6\webapps\BOE\WEB-INF\internal\global.properties

Changing cache-control to 10 years might seem extreme, but used in conjunction with a "Last-modified" property, it still allows for conditional revalidation of the cache. The Last-modified property is used extensively in SAP BI Platform 4.0.



Info about conditional revalidation of cache is described in the RFC Specification for HTTP Caching as:

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.

Between using HTTP compression by default, and the addition of consistent Cache-Control headers, SAP BI Platform 4.0 has some compelling features that immediately help improve user experience with the web applications. In the next section, we'll discuss some challenges and limitations that still exist and how we might address them.

Challenges and Limitations

Based on the enhancements to 4.0, it may seem that performance need not be a concern for administrators. However, there are still some considerations that must be taken into account to ensure the application performs as well as possible. The considerations addressed in this document are:

- Experience for new users, or users who have recently cleared their browser cache
- Performance implications with Web Caches such as proxies and SSL
- Impact of 64-bit JVM on application server performance and garbage collection

The overarching challenge to keep in mind is that BI Launch Pad is a large web application, and anything we can do to reduce the bytes transferred to a client, or the volume of requests sent to the application server, will have a positive knock-on effect to performance.

Users without cache

For any user without existing cache for BI Launch Pad, whether they are brand new users or users who have cleared their browser cache for any number of reasons, there can be a notable performance lag when accessing the portal for the first time. This has to do with the number of JavaScript, and supporting files that comprise the user interface (UI) of BI Launch Pad.

For example, the process of logging in, accessing the document list, and navigating to the 'Web Intelligence Samples' folder sends nearly 1MB of files over the wire. Using Fiddler, we can note the following breakdown:

```
Request Count:      251
Bytes Sent:   254,235      (headers:251068; body:3167)
Bytes Received: 905,636    (headers:65032; body:840604)

ACTUAL PERFORMANCE
--------------
Requests started at:       18:29:20.372
Responses completed at:    18:31:14.076
Sequence (clock) time:     00:01:53.7031250


RESPONSE BYTES (by Content-Type)
--------------
 text/javascript:   656,335
       ~headers~:   65,032
       image/gif:   53,339
        text/css:   44,411
       image/png:   38,295
       text/html:   30,296
       text/json:   14,931
application/json:   2,997
```

In summary, 251 total requests, nearly 1MB of data, with the entire process taking almost 2 minutes to complete. You can see that ~73% of the total size is found in a number of JavaScript files.

Now, compare this with the access of a user who has already cached the foundation of BI Launch Pad on their local browser:

```
Request Count:      22
Bytes Sent:   25,431 (headers:22264; body:3167)
Bytes Received: 53,699     (headers:5585; body:48114)

ACTUAL PERFORMANCE
--------------
Requests started at:       18:53:37.107
Responses completed at:    18:54:29.529
Sequence (clock) time:     00:00:52.4218750


RESPONSE BYTES (by Content-Type)
--------------
       text/html:   30,201
       text/json:   14,916
       ~headers~:   5,585
application/json:   2,997
```

The same process now requires 22 total requests, a mere 50KB of data, with the process taking around 50 seconds. The remaining requests are either marked with a Cache-Control header of 'no-cache' which forces the client to request a new copy from the origin server, or they are executed with a POST which requires server interaction. These requests are the dynamic content that make of the BI Launch Pad experience.

By virtue of this simple test the importance of caching becomes abundantly clear. To maximize the user experience, it is critical that web caching be as pervasive as possible. Don't forget, this is static content … content that does not change for each user, and that is completely re-usable until the content is changed by an SAP patch. Applying a patch updates the "Last-modified" time stamp and allows us to refresh cache as necessary.

Proxies and SSL as factors in performance

There are 2 additional components that play a role in the distribution of cache as it pertains to SAP BI Platform 4.0 web applications.

In addition to browser caching, the HTTP protocol allows proxy caching as well, which enables static resources to be cached on corporate web proxy servers. It ensures that even first time users of BI Launch Pad can benefit from caching. This is especially helpful in large deployments where users may be accessing in locations far from where the application servers reside. Most large businesses have proxy servers in place, and users often don't even realize they are accessing via a proxy. For example, I access the SAP corporate network through a proxy server when I VPN in from home to work late on this documentation.

In order to benefit from proxy caching, resources should be marked with a 'Cache-Control: public' header so that proxies know it can be cached and provided freely to other users requesting the same resource.

To illustrate the importance of leveraging proxy cache, consider an example where a large corporation has offices in the Western US, Germany, and China. The SAP BI Platform servers are hosted in the Western US location. By monitoring the same BI Launch Pad workflow identified in the previous section, Fiddler can give us the following summary:

```
US West Coast
---------------
Elapsed Time:  61.10s


China
---------------
Elapsed Time:  141.95s


Japan / Northern Europe
---------------
Elapsed Time:  72.65s
```

You can see that while users in Germany only suffer a ~15% slower performance, users in China might suffer by over 100%! Fiddler admittedly offers very rough estimates but for the purposes of illustration it is still valuable. The point is that users in other geographical regions can potentially suffer a significant decrease in performance from an application hosted far away. By seeding proxy cache whenever possible we can reduce the potential performance hit significantly.

Unfortunately, proxy cache cannot be leveraged when SSL is in use. Proxy servers are unable to cache (or interpret) SSL content so the benefits of using proxy cache are not available. However, some older browsers do not properly cache files delivered via SSL. To ensure all browsers correctly cache resources transferred over SSL, it is important to use the 'Cache-Control: public' header on the relevant content.

Google has a great overview on Optimizing Caching that addresses these and other helpful concepts. I will refer back to this resource often throughout the rest of the document.

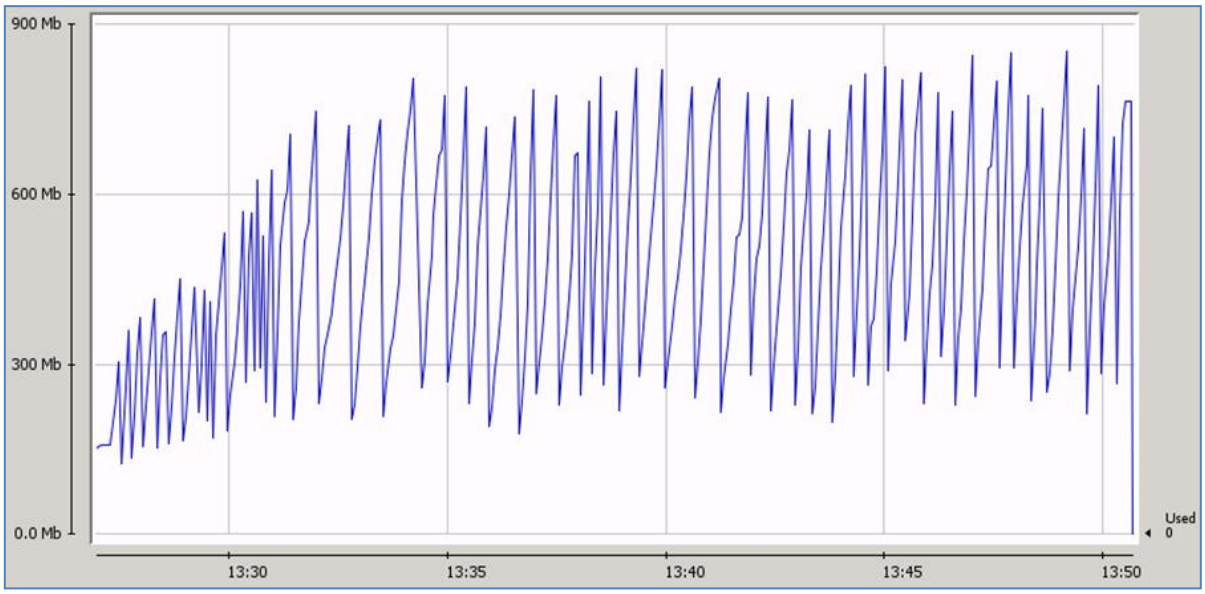
64-bit JVM and Garbage Collection

Although SAP BusinessObjects Enterprise XI 3.1 supports the use of 64-bit JVM for application server use, the usage of such JVM is expected to increase dramatically in SAP BI Platform 4.0 due to the fact that Tomcat 6 is now bundled with the application and ships with a 64-bit JVM by default.

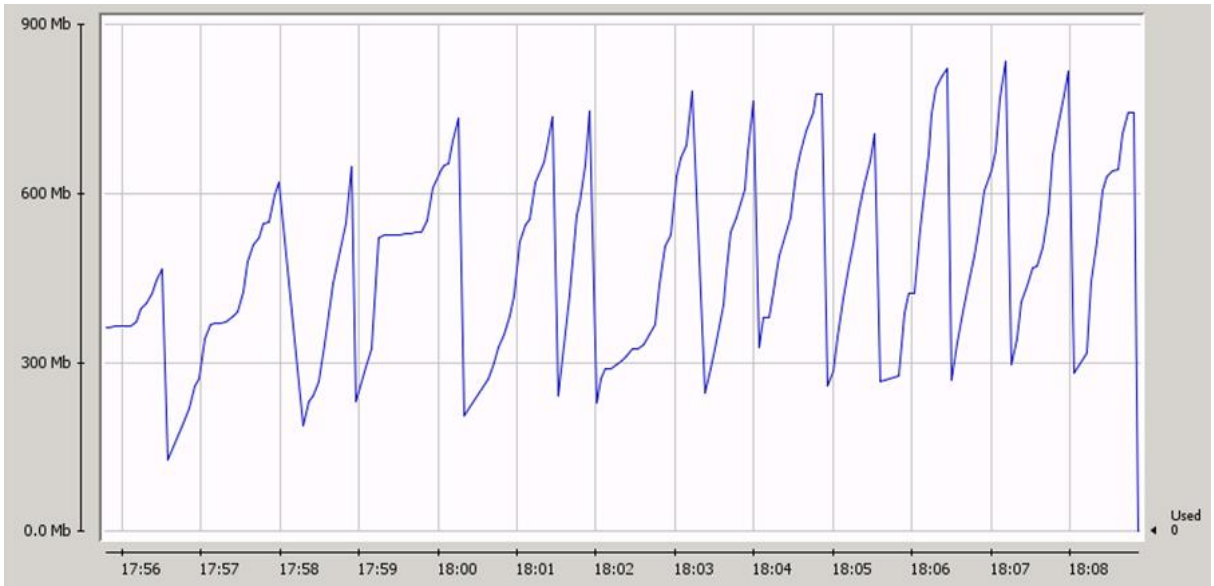
In general larger heap sizes mean a reduction in pesky out of memory errors, and the ability to service more simultaneous requests on a single application server instance. However, it also entails a potential for large increases to garbage collection with a larger heap for dead or expired objects to accumulate in. Overly frequent garbage collection can result in pause times to the application and diminished throughput due to significant CPU cycles spent on reclaiming heap space. While this document won't act as a tutorial for garbage collection, there are numerous overviews available. The primary thing to keep in mind about garbage collection in this document is that the less often it occurs, the better the application server will perform.

In a simple performance test executed with 50 users following the workflow mentioned above, the following memory statistics were collected via JConsole.

Tomcat alone required 95 collections:



The same test executed with Apache in front of Tomcat with static content deployed separately. Here Tomcat required only 48 collections:



We will discuss these results in greater detail below, along with the parameters used on Apache to impact the greatest change. For now, simply note that the Apache example demonstrates fewer spikes in memory. By separating static and dynamic content, and reducing the volume of requests that Tomcat needs to serve, memory usage by the application server becomes more normalized. In summary, we let Tomcat do what it does best, process dynamic content such as JSP and servlet requests, and enable an HTTP server like Apache to cover the remaining static content.

Proposal Framework

Now that we've discussed some of the challenges to performance in SAP BI Platform 4.0, we can outline a specific proposal for addressing them. The result of this proposal is architecture with the following components:

- Web Server - Apache 2.2.19 + mod_jk 1.2.32 + Mod_headers, mod_deflate, mod_cache
- Application Server - Tomcat 6.0.24 + JDK 1.6.0_26
- Deployment - WDeploy executed in split mode in order to separate static and dynamic content

I will discuss each of the components listed above in detail and explain how they contribute to a better user experience. I will also walk through the configuration of each area, Web Server, Application Server, and Deployment to enable an administrator to set up this proposal from end to end. Keep in mind that every environment is different, and there may be environmentally specific items that require some of these steps be modified for use in your own deployment. I will try to provide enough background on the components so that the proposal can be modified as necessary.

Though I will outline a solution using Apache and Tomcat, it is possible to deploy the same functionality using other application server vendors such as IBM HTTP Server and WebSphere.

Once we've covered each topic, I will provide a walkthrough of how to configure the proposed environment and confirm it is functional.

Web Server

We've discussed Apache already, so we know it is an open source HTTP server. Apache can be downloaded from:

http://httpd.apache.org/download.cgi

Apache is updated regularly but at the time of writing the latest version was 2.2.19. Apache is available for numerous platforms and, as open source software, can be downloaded as source code or as a compiled binary (Windows only). On Windows, you can download an MSI with or without SSL support. Since enabling SSL is optional, I suggest downloading the version with SSL support to simplify things if you decide to use SSL down the road. The appropriate Windows download name is:

```
Win32 Binary including OpenSSL 0.9.8r (MSI Installer)
```

Apache must be installed prior to moving on to any additional web server related tasks. It can be installed with all of the default options for testing this configuration.

Mod_jk

Mod_jk is the Apache specific AJP13 connector that allows HTTP servers such as Apache to communicate with Tomcat. It allows Apache to become a front end to host static content, perform load balancing, or act as a reverse proxy and has a wealth of other features and functionality. There is a plugin for IBM HTTP Server (IHS) that delivers comparable functionality for use with IBM WebSphere.

The excellent Tomcat Connectors documentation contains references for the configuration files needed to set up an AJP13 connector and reference guides for a number of web servers (Apache, IIS, SunOne, etc). There is also a 'for the impatient' walkthrough that I will use to deploy the solution. At the time of writing, the latest version of mod_jk was 1.2.32 and could be downloaded from the Tomcat Connectors website above.

Mod_jk is a shared object module (.so file) that resides in the modules directory of Apache. It is configured with worker threads that handle the forwarding of requests from Apache to Tomcat. Apache knows which requests to forward based on uniform resource identifier maps, or URI maps. A URI map takes this format:

```
JkMount /BOE/*/PlatformServices/*.object ajp13
```

JkMount(s) are added in either the httpd.conf file for Apache, or in external files that are referenced in the httpd.conf via include statements. A complete example will be demonstrated below. This means that Apache maps (or mounts) everything under the directory PlatformServices that ends in an extension of .object to a worker thread named 'ajp13'. When a request is received by Apache for:

```
/BOE/portal/1107111813/PlatformServices/service/app/logon.object
```

It knows based on the JkMount command to send this request to Tomcat via a worker named 'ajp13' for processing.

Workers are defined in a file called workers.properties. A true production workers.properties is often fairly complicated should load balancing or other advanced parameters need defining. For the purposes of this document though, we will keep the configuration very simple. A simple workers.properties file might look something like this:

```
# Define 1 real worker using ajp13
worker.list=ajp13
# Set properties for worker1 (ajp13)
worker.ajp13.type=ajp13
worker.ajp13.host=localhost
worker.ajp13.port=8009
```

This configures a single worker, named 'ajp13'. The worker uses the ajp13 protocol and communicates with Tomcat on the address localhost:8009. We'll configure the AJP connector port on Tomcat in the next section.

In summary, we implement mod_jk to enable a bridge between Apache and Tomcat that allows us to host static content on the web server and forward any requests for dynamic content to the application server. Remember, mod_jk is a robust technology with many fine-tuning options available. This example uses an extremely simple setup that is probably not sufficient for production use but does provide a functional environment.

Mod_headers

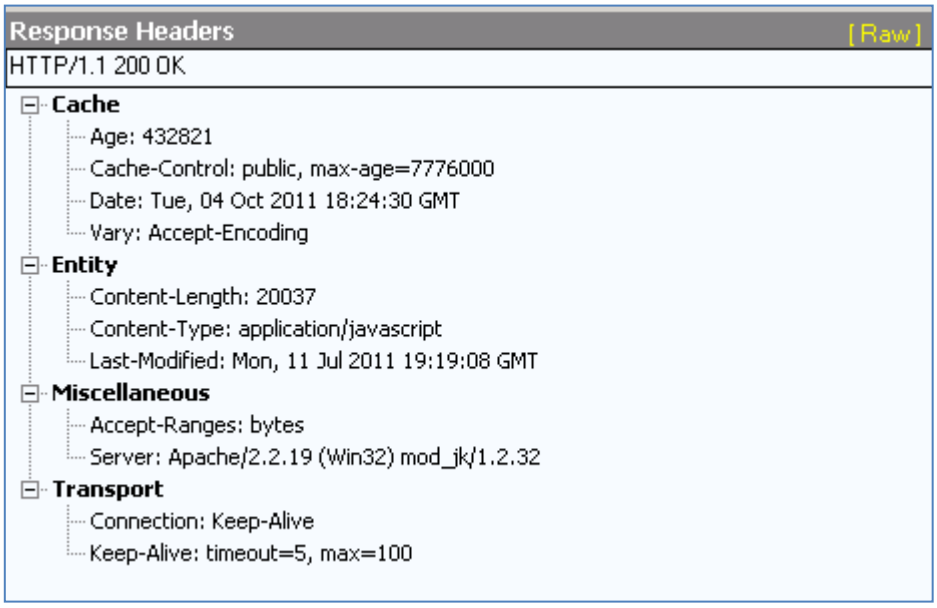
Mod_headers is a module that allows you to control and modify HTTP request and response headers and is part of the standard Apache package. This is a critical component that allows us to apply valuable Cache-control headers like 'public' and 'max-age' to static BI Launch Pad resources hosted on our Apache server. The very thorough mod_headers documentation can be found here. If you refer to the images on Pages 3 & 4 you can see what the headers on a piece of static content (utils.js) look like out of the box. Note that it does not contain a 'public' Cache-control header. It also contains both a 'Last-modified' header as well as an ETag, which the optimizing cache document tells us is superfluous:

It is important to specify one of Expires or Cache-Control: max-age, and one of Last-Modified or ETag, for all cacheable resources. It is redundant to specify both Expires and Cache-Control: max-age, or to specify both Last-Modified and ETag.

By using the mod_headers module, we're able to improve the cache-ability of our static resources by easily modifying Cache-control and removing either the Last-modified or ETag response header. We'll discuss the pros and cons of using either one below. The resulting request/response would look like this:

enterprise	/BOE/portal/1107111813/InfoView/logon.jsp	2,356	
enterprise	/BOE/portal/1107111813/InfoView/common/appService.do?service=skinning&resource=styles...	3,847	max-age=315360000
enterprise	/BOE/portal/1107111813/InfoView/js/utils.js	5,376	public, max-age=7776000
enterprise	/BOE/portal/1107111813/shared/js/caf/helpSystem.js	1,292	public, max-age=7776000
enterprise	/BOE/portal/1107111813/InfoView/help/en/html/HelpFileMap.js	193	public, max-age=7776000

*In this example, I've reduced the validity of the static resource to **7776000** seconds or 90 days to illustrate the difference between a resource served by Tomcat (such as appService.do) and a static resource. The response headers show the complete Cache-control header and the ETag has been removed.*



Mod_headers can perform this type of action on a wide variety of response and request headers and the configuration is limited to a few lines in the httpd.conf file. It even carries a useful limitation in that it won't affect headers on files served by Tomcat. SAP already correctly sets the 'no-cache' response header on dynamic content such as the contents of the folder list in BI Launch Pad. We don't want Apache to supersede any of the legitimate headers that are configured by default on dynamic content.

ETag vs. Last-modified

An Entity Tag, or ETag is one of several mechanisms that HTTP provides for cache validation. It allows a client to make conditional requests. This allows caches to be more efficient, and saves bandwidth, as a web server does not need to send a full response if the content has not changed.

ETags, and the Last-modified property, specify some characteristic about the resource that the browser checks to determine if the files are the same. In the Last-Modified header, this is always a date. In the ETag header, this can be any value that uniquely identifies a resource. An ETag might look something like this:

```
W/"2762-1310426288169"
```

These headers allow the browser to efficiently update its cached resources by issuing conditional GET requests when the user explicitly reloads the page. Conditional GETs don't return the full response unless the resource has changed at the server, and thus have lower latency than full GETs.

There are limitations to each header, but as mentioned previously, only one of the two options must be used to provide conditional revalidation of cache.

In the case of Last-modified, the optimizing cache walkthrough has this to say:

Last-Modified is a "weak" caching header in that the browser applies a heuristic to determine whether to fetch the item from cache or not. (The heuristics are different among different browsers.)

This simply means that each browser (Firefox, Internet Explorer, etc) makes a determination of whether or not to request the complete resource from the server and thus there is the possibility that the resource could be fetched prematurely. This seems a weak deterrent to using Last-Modified, however, since the value is updated upon applying an SAP patch where the file has been changed.

The limitation with ETags when used in a default configuration of Apache or Tomcat is that they are intended to be used on a single server. The ETag therefore contains some information that is specific to the origin server's hostname. In deployments with multiple web/application servers ETags can cause unnecessary load by serving the same resource from different servers, even though the resource is identical and unchanged.

Fortunately Apache has a solution to this problem. Mod_headers gives us the ability to specify that ETags be created without the hostname as a component. I'll outline this method below, but an excellent tutorial on the process also exists here.

Ultimately, it matters little whether you use Last-Modified or ETag and I'll demonstrate how to use either approach. It is important to raise awareness on this topic though so that we choose an option that maximizes our ability to disperse and persist cache to client browsers.

Mod_headers provides a great deal of flexibility in normalizing Cache-control for BI Launch Pad resources. It can be used to achieve maximal caching at both the browser and proxy cache levels and has a great impact on overall performance.

Mod_deflate

We won't spend much time on mod_deflate, as it is simply providing the HTTP compression that the default configuration of SAP BI Platform 4.0 provides with Tomcat. This module is also available in the default distribution of Apache and requires no additional download. The documentation for mod_deflate provides an excellent example that I'm using "as is" for this document. The AJP connector on Tomcat does not support compression, so it is critical that we configure it at the web server level so the benefits of reduced footprint are not lost.

Mod_cache

We've now discussed several mechanisms for improving the user experience in BI Launch Pad but **mod_cache** might be the most compelling of them all. In short, mod_cache implements a content cache that can be used to cache static resources served by Apache, as well as any remaining cacheable resources served by Tomcat. It can be implemented as either a disk, or file system based cache (mod_disk_cache), and as an in-memory cache (mod_mem_cache). Because Apache keeps an index of the cache, it is still significantly faster to serve a resource from cache than it is to traverse the htdocs directory and retrieve the file each time it is requested. When configured properly, mod_cache allows us to maintain a cached copy of all static resources that comprise the BI Launch Pad. Mod_cache is included in the default distribution of Apache and does not need to be downloaded separately.

In this proposal we'll use mod_disk_cache due to some key areas I will discuss below. I have tested mod_mem_cache as well, and while it works, it is not as effective as mod_disk_cache.

There are several guides that discuss the use of mod_cache in great detail and I highly recommend familiarizing yourself with them if you choose to use this module in conjunction with SAP BI Platform 4.0. The guides are:

Apache Caching Guide

Htcacheclean

Htcacheclean is the tool Apache provides to automate cleanup of file based caches such as mod_disk_cache

Mod_cache is an excellent fit for use with SAP BI Platform 4.0 because it respects several key features regarding cache-ability:

1. The request must be a HTTP GET request. This means that dynamic data processed with POST requests, such as navigation through the folder view in BI Launch Pad, are not cached.

2. If the response contains an "Authorization:" header, it must also contain a "public" option in the "Cache-Control:" header. This means that mod_cache respects that responses requiring Kerberos or NTLM should not be cached by default.

3. If the response has a status of 200 (OK), the response must also include at least one of the "ETag", "Last-Modified", or the max-age directive of the "Cache-Control:" header. This ensures mod_cache does not serve up stale resources to client requests.

4. Correctly revalidates cache entities which have Cache-Control: no-cache set in their response headers. This ensures that we don't cache dynamic data incorrectly by always checking with Tomcat.

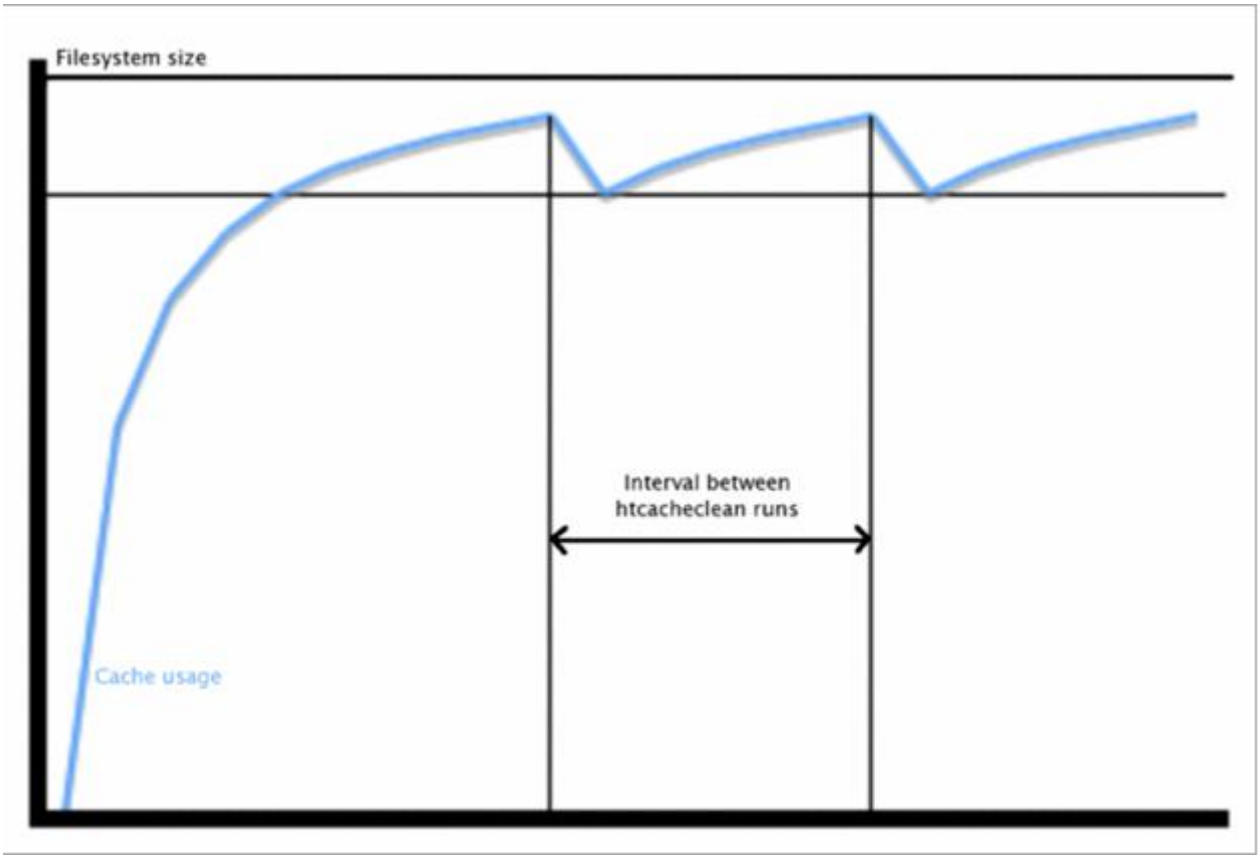
Mod_disk_cache vs. mod_mem_cache

Mod_disk_cache is the mod_cache mechanism I have chosen to implement for a few reasons:

1. Disk space is generally a cheaper resource than memory.
2. Disk cache can be reused following a restart of Apache. Restarting Apache removes in-memory cache.
3. Mod_mem_cache is per process, which means that Apache on platforms where multiple child processes are created (Unix/Linux) has the potential to use high amounts of memory to enable caching

There are some security implications around using mod_disk_cache. Please refer to the Security Considerations section of the Caching Guide for info on how to address these. Most of these can be accounted for by running Apache as a user account with limited privileges, that is, rights to the Apache and cache directories only.

With the addition of Htcacheclean to manage the size of the disk cache we have a sustainable long term approach to caching. This tool can be run in unattended mode, so that it periodically deletes old cache entries and keeps the overall size of the cache directory in check.



Remember that mod_cache is a powerful mechanism that effectively provides you with a built in reverse proxy. Requests will be served by the caching module unless it determines that the backend should be queried. This document will attempt to explain the usage of mod_cache but, as every implementation is different, it should be tested carefully before applying its use to a Production environment.

We've now covered all of the Apache modules used in this framework. Apache provides many other modules that extend its functionality and make it one of the most powerful engines available to us for improving the user experience in SAP BI Platform 4.0.

Application Server

Tomcat

SAP BI Platform 4.0 bundles Tomcat 6.0.24 as the default java application server for the deployment. At time of writing, the latest version of Tomcat 6 was 6.0.33. It is recommended that you consider taking the latest update to Tomcat in order to benefit from the 9+ releases worth of bug fixes released by Apache. SAP Note 1497173 has the details of how to perform a simple upgrade of Tomcat. The steps are specifically for XI 3.1 but are applicable to 4.0 as well.

In order for Tomcat to communicate properly with Apache and mod_jk we need to enable an AJP connector in the server.xml. The AJP connector receives requests from the mod_jk plugin and responds accordingly so that Apache can serve the resources. By default the AJP connector uses a listening port of 8009. As indicated previously, the AJP connector does not support HTTP compression (AJP is a different protocol from HTTP so this makes sense) so the configuration is fairly straightforward. I will cover the exact steps below but a configured AJP connector might look like this:

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" URIEncoding="UTF-8"
enableLookups="false" connectionTimeout="20000"/>
```

JDK

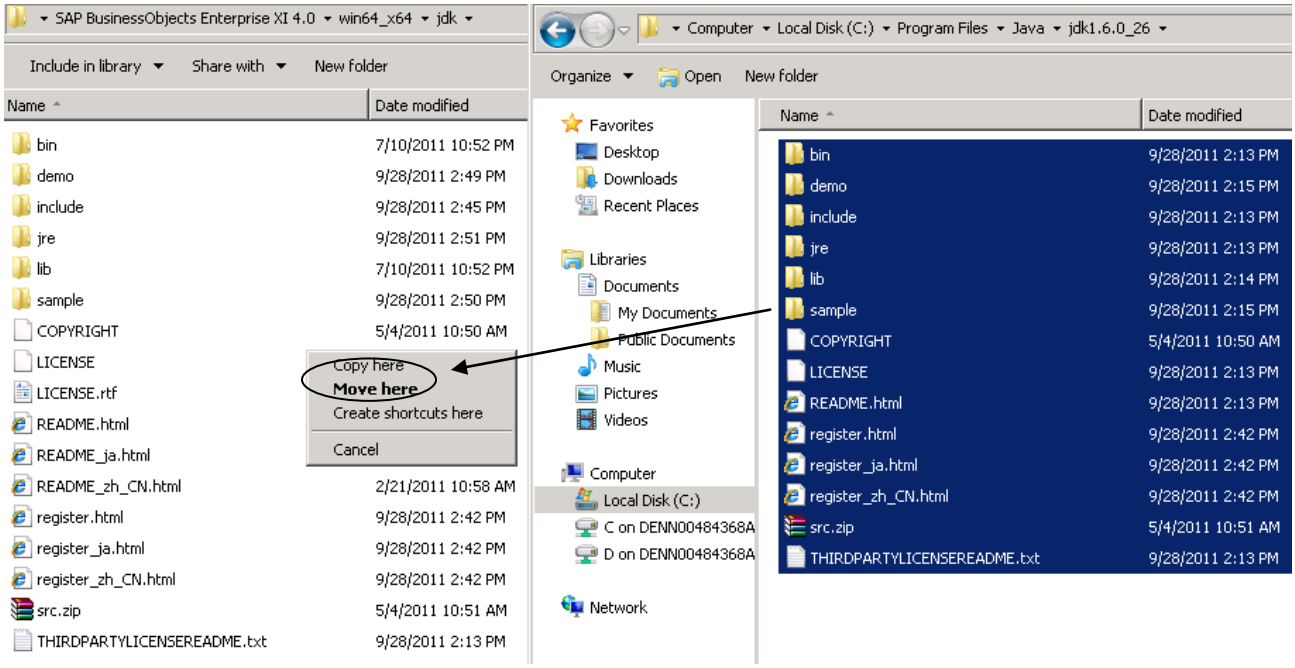
As with Tomcat, the Sun JDK has been updated several times since the release of SAP BI Platform 4.0. It is recommended that you update the JDK that Tomcat uses to version 1.6.0_26, at a minimum. You can download Sun JDK 1.6.0_26 from the following location:

http://www.oracle.com/technetwork/java/javase/downloads/jdk-6u26-download-400750.html

Stop Tomcat, install the JDK, and replace the binaries under the following directory:

```
Windows:[InstallDir]\ SAP BusinessObjects Enterprise XI 4.0\win64_x64\jdk

Unix: [InstallDir]\sap_bobj\enterprise_xi40\[Platform64]\jdk
```



The directory structure is identical, so there should be no need to update any properties in Tomcat. Restart the application server after you're finished.

We've now covered the basic premises related to the application server in this proposal. Since we now have Apache in the equation to do much of the heavy lifting, we can lighten the load on Tomcat and allow it to process the dynamic content of BI Launch Pad as effectively as possible.

WDeploy

The final piece in the puzzle is the execution of WDeploy to separate the static and dynamic content, configure the URI mapping for mod_jk and deploy static content to the Apache 'htdocs' directory. It also handles the deployment of the .war files to Tomcat. The process is also covered in SAP Note 1325139

This example will use a configuration where Apache and Tomcat reside on the same machine. It is possible to separate the two and manually deploy the static content to Apache but that particular process will not be described here.

The prerequisites for running WDeploy are to make sure the tool knows where to find the relevant resources for Apache and Tomcat. These files, prefaced by 'config.<servername>', tell WDeploy the install directories and where the content should be deployed.

In a default Windows installation, the files are located at:

```
C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI
4.0\wdeploy\conf
```

Open config.apache in a text editor and set the parameter 'ws_dir' to the root of the Apache install like this:

```
ws_dir=C:\Program Files (x86)\Apache Software Foundation\Apache2.2\
```

Save and close the file.

Next, open the file config.tomcat6 and make sure the first 3 entries in the file look like this:

```
# as_dir: the installation directory of the application server
#as_dir=/home/qaunix/Tomcat6018
as_dir=C:\Program Files (x86)\SAP BusinessObjects\Tomcat6

# as_instance: the application server instance to deploy to (represents the name of a
folder in the conf/Catalina directory)
as_instance=localhost

#on windows, if tomcat is installed as a service, specify
#service name to modify service startup parameters too
as_service_name=BOEXI40Tomcat
```

Save the file and close it.

Now, with all of the prerequisites in place, we can move on to the actual implementation of this proposal. I'll provide step-by-step instructions for configuring each of the elements discussed above and outline key troubleshooting steps to determine whether the environment is set up correctly and is having the desired effect.

Configuration

Setup of the environment starts with 3 assumptions:

1. Apache has been installed with all of the default options and is listening on HTTP port 80 (if you have IIS installed on the machine make sure it is stopped)
2. SAP BI Platform 4.0 has been installed to the default location along with Tomcat 6.0.24
3. The environment is Windows based. Note that Unix/Linux platforms support the same configuration but that some resources (such as mod_jk) will need to be compiled from source. That process is not covered in this document.

The specific steps covered here will be:

1. Configure Apache/Tomcat and modules including mod_jk
2. Run WDeploy
3. Test

Configuring Apache

You can test the default installation of Apache by launching a browser and typing the URL:

http://localhost

If Apache is running you should get a page that says, "It works!"

I've commented the entries in each configuration file but will also add context when necessary.

Mod_jk

1. Download the most recent version of mod_jk.so from the Tomcat Connectors Download site
    a. At the time of writing, version 1.2.32 could be downloaded here
2. Copy the file mod_jk.so to C:\Program Files (x86)\Apache Software Foundation\Apache2.2\modules
3. Navigate to `C:\Program Files (x86)\Apache Software Foundation\Apache2.2\conf\extra` and create a file called `httpd-BI40.conf`
4. Open the file in a text editor such as notepad

5. Add the following lines to the top of the file:

```
#====================LoadModules=======================
LoadModule jk_module modules/mod_jk.so



#====================Configure mod_jk==============
# Where to find workers.properties
# Update this path to match your conf directory location (put workers.properties
next to httpd.conf)
JkWorkersFile conf/workers.properties
# Where to put jk shared memory
# Write shared memory to the logs directory
JkShmFile logs/mod_jk.shm
# Where to put jk logs
# Update this path to match your logs directory location (put mod_jk.log next to
access_log)
JkLogFile logs/mod_jk.log
# Set the jk log level [debug/error/info]
JkLogLevel    info
# Select the timestamp log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"
```

Save and close the file.

6. Next, create the workers.properties file in the C:\Program Files (x86)\Apache Software Foundation\Apache2.2\conf directory and paste the following into the file:

```
# Define 1 real worker using ajp13
worker.list=ajp13
# Set properties for worker1 (ajp13)
worker.ajp13.type=ajp13
worker.ajp13.host=localhost
worker.ajp13.port=8009
```

Save and close the file.

7. Navigate to the directory C:\Program Files (x86)\SAP BusinessObjects\Tomcat6\conf and make a backup of the file server.xml
Open server.xml in a text editor such as notepad and comment in the AJP connector section like this:

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
 <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" URIEncoding="UTF-8"
enableLookups="false" connectionTimeout="20000"/>
```

Save and close the file.

8. Restart Apache and Tomcat to enable the mod_jk connector

Mod_headers

Mod_headers is included with the distribution of Apache 2.2 so it does not need to be downloaded separately.

1. Navigate to `C:\Program Files (x86)\Apache Software Foundation\Apache2.2\conf\extra` and open `httpd-BI40.conf` in a text editor
2. Add the following line under the LoadModules section so it looks like this:

```
#====================LoadModules=======================
LoadModule jk_module modules/mod_jk.so
LoadModule headers_module modules/mod_headers.so
```

3. Add a new section below 'Configure mod_jk' that looks like this to use **Last-Modified only**:

```
#====================Configure mod_headers==============
FileETag None
Header unset ETag

   <FilesMatch "\.(gif|jpe?g|png|html?|js|css)$">
               Header set Cache-Control "public, max-age=315360000"
   </FilesMatch>
```

   This tells Apache to remove the ETag header from any requests that contain it, and to set the header Cache-Control for any static resources (images, html, JavaScript, and style sheets) so it contains both public and max-age settings.

4. To use **hostname agnostic ETags** instead, add the following lines:

```
#====================Configure mod_headers==============
FileETag MTime Size
Header unset Last-Modified

   <FilesMatch "\.(gif|jpe?g|png|html?|js|css)$">
               Header set Cache-Control "public, max-age=315360000"
   </FilesMatch>
```

   This tells Apache to create ETags based only on mime type and file size and eliminate the hostname dependency. The rest of the command is the same.

Mod_deflate

Mod_deflate is included with the distribution of Apache 2.2 so it does not need to be downloaded separately.

1. Navigate to `C:\Program Files (x86)\Apache Software Foundation\Apache2.2\conf\extra` and open `httpd-BI40.conf` in a text editor
2. Add the following line under the LoadModules section so it looks like this:

```
#====================LoadModules=======================
LoadModule jk_module modules/mod_jk.so
```

```
LoadModule headers_module modules/mod_headers.so
LoadModule deflate_module modules/mod_deflate.so
```

3. Add a new section below 'Configure mod_headers' that looks like this:

```
#====================Configure mod_deflate==============
AddOutputFilterByType DEFLATE text/html text/xml text/plain text/javascript
text/json application/javascript text/css
#Highest 9 - Lowest 1
DeflateCompressionLevel 9

# Don't compress images
SetEnvIfNoCase Request_URI \
       \.(?:gif|jpe?g|png)$ no-gzip dont-vary


#Optional
#Logging
DeflateFilterNote ratio
LogFormat '"%r" %b (%{ratio}n) "%{User-agent}i"' deflate
CustomLog logs/deflate_log.log deflate
```

Save the file and exit.

The entry, AddOutputFilterByType tells Apache which MIME types it should compress. The list reflects the default settings that SAP now configures in the Tomcat server.xml for the HTTP connector. We don't compress images because the standard formats are already very well compressed. It isn't worth the CPU cycles to compress them anymore.

Mod_cache

The configuration of mod_cache will comprise both the settings required to enable the feature, as well as some sample command line parameters for running htcacheclean.

1. Navigate to C:\Program Files (x86)\Apache Software Foundation\Apache2.2\conf\extra and open httpd-BI40.conf in a text editor
2. Add the following line under the LoadModules section so it looks like this:

```
#====================LoadModules=====================
LoadModule jk_module modules/mod_jk.so
LoadModule headers_module modules/mod_headers.so
LoadModule deflate_module modules/mod_deflate.so
LoadModule cache_module modules/mod_cache.so
LoadModule disk_cache_module modules/mod_disk_cache.so
```

3. Add a new section below 'Configure mod_deflate' that looks like this:

```
#====================Configure mod_cache==============
<IfModule mod_cache.c>

#Address the Thundering Herd identified in Apache's Caching Guide
CacheLock on
CacheLockPath C:/temp/mod_cache-lock
CacheLockMaxAge 5
```

```
#This parameter tells Apache to ignore unique session identifiers when caching
#static content.  SAP BI Platform 4.0 uses the strings jsessionid and bttoken to
#identify user sessions.
CacheIgnoreURLSessionIdentifiers jsessionid bttoken
#Don't cache cookies as they are unique per user
CacheIgnoreHeaders Set-Cookie

#Enable mod_disk_cache instead of mod_mem_cache
<IfModule mod_disk_cache.c>

CacheRoot c:/cacheroot
CacheEnable disk /
CacheDirLevels 2
CacheDirLength 1

</IfModule>
</IfModule>
#====================End httpd-BI40.conf==============
```

Save and close the file

The thundering herd is a concept covered in the mod_cache documentation, but the summary is:

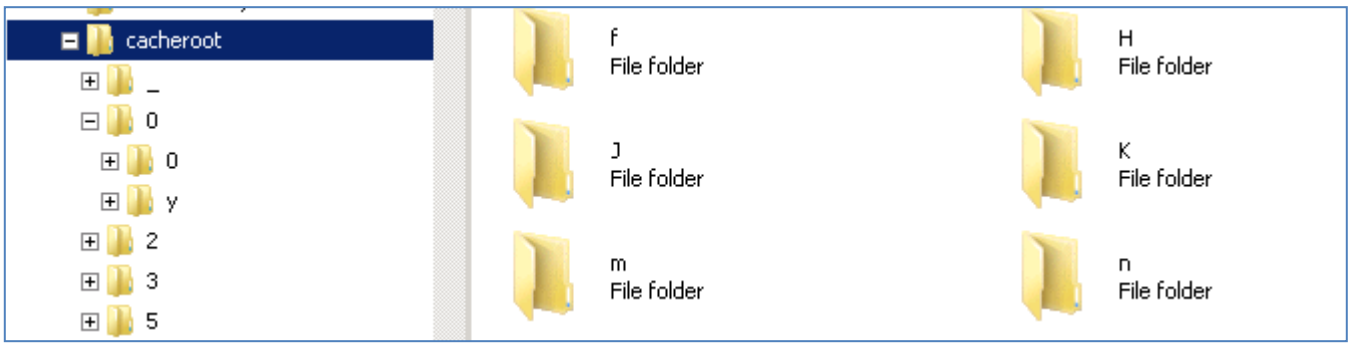
When a cached entry becomes stale, mod_cache will submit a conditional request to the backend, which is expected to confirm whether the cached entry is still fresh, and send an updated entity if not.

A small but finite amount of time exists between the time the cached entity becomes stale, and the time the stale entity is fully refreshed. On a busy server, a significant number of requests might arrive during this time, and cause a thundering herd of requests to strike the backend suddenly and unpredictably.

We combat the herd by using the CacheLock settings for mod_cache. These are important settings to leverage, especially in very busy environments. Lock files, which tell clients that a stale resource is currently being refreshed, will be written to the directory specified in CacheLockPath.

4. To configure the CacheLock mechanism, create the folder identified in 'CacheLockPath', in this case C: /temp/mod_cache-lock, and make sure the Apache user has Read/Write privileges to the location.
5. The 'CacheRoot' parameter is where Apache will write the cache files that are served to users. Create the directory, in this example C:/cachroot, and ensure the Apache user has Read/Write privileges to the location
6. The 'CacheEnable' parameter tells Apache what to cache, in this case, we're caching everything that meets the critieria for caching to disk. Other examples can be found in the mod_cache documentation here.
7. The 'CacheDirLevels' and 'CacheDirLength' directives configure how many sub-directory levels are used to store cache, and how many characters can be used in the folder names. It is recommended you keep these numbers low, as with values of 5 and 4 respectively it is possible to have millions of files and folders, making maintenance very challenging.

A cache root directory configured with the values in the above example might look like this:



That is, a series of folders 2 levels deep with only 1 character. It is easy to see what the impact could be of having 5 levels of directories with 4 characters in the name!

Htcacheclean

Htcacheclean is a tool provided by Apache that allows us to keep the size of the mod_disk_cache 'CacheRoot' directory within a certain limit. The cache itself is not intelligent, so it is important to use this tool so that the size of the directory does not get out of hand. The tool can be run manually, or in unattended mode (daemon mode). I will demonstrate both methods below but highly recommend unattended mode for use in Production environments. This ensures that the cache is cleaned on a regularly scheduled interval.

Htcacheclean can be found in the 'bin' directory of the Apache installation. In a default Windows installation, this would be C:\Program Files (x86)\Apache Software Foundation\Apache2.2\bin. We'll run the utility from a command prompt a couple of times to provide background on how the tool works and then create a Windows Task to handle the automation every time the machine starts. On Unix/Linux you might use cron to automate the htcacheclean process.

Htcacheclean from a command prompt

1. Launch a command prompt (Start > run > cmd)
2. `cd C:\Program Files (x86)\Apache Software Foundation\Apache2.2\bin`

3. Run the following command:

   ```
   htcacheclean.exe -D -p C:/cacheroot -l 50M -v
   ```

   The command should return output like this:

   ```
   Statistics:
   size limit 50.0M
   total size was 602.6K, total size now 602.6K
   total entries was 105, total entries now 105
   ```

   Running htcacheclean with the –D parameter tells the utility to perform a dry run where nothing is actually deleted. The –p parameter specifies the location of the cache, -l specifies a target size for the entire cache directory, and –v causes statistics to be printed, as above.

4. In order to actually clean the cache from the command line use the following syntax:

   ```
   htcacheclean.exe -D -p C:/cacheroot -l 1M -v
   ```

   The command should return output like this:

```
Statistics:
size limit 1.0M
total size was 1.1M, total size now 1017.3K
total entries was 211, total entries now 179
```

This command specified a size limit of 1MB and you can see the cache was reduced by a total of 32 entries.

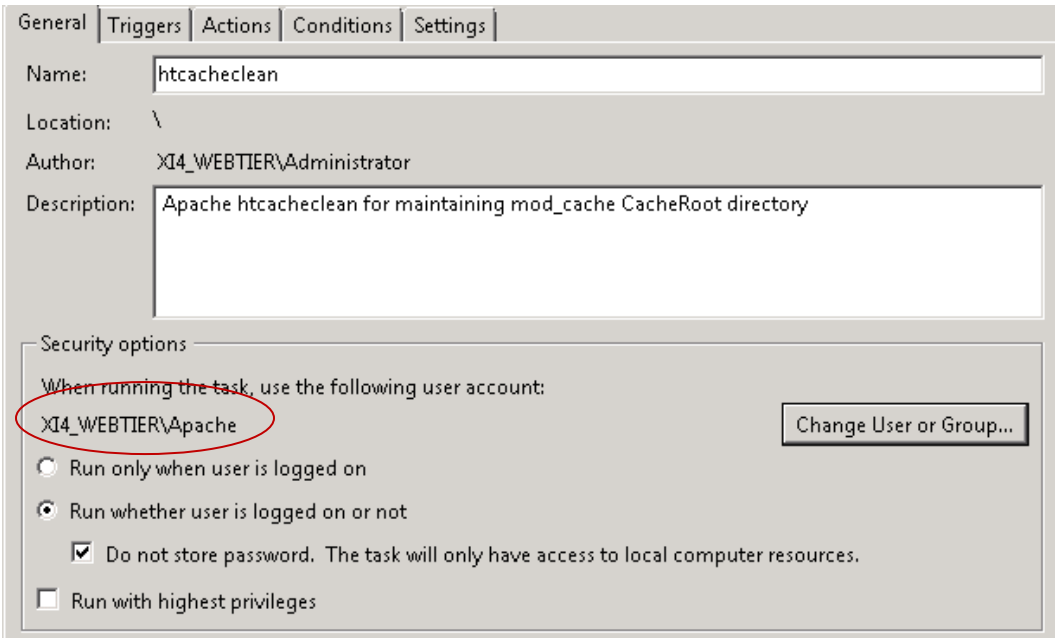
Htcacheclean in unattended mode

Using the command line is a great method for testing the right parameters. Before configuring htcacheclean to run as a scheduled task, we'll need to decide how large we want to let our cache grow. With the setting of "2" used for CacheDirLevels in the above example, a total of 4096 subdirectories can ultimately be created. In conjunction with the hashing mechanism mod_cache uses to create cached files, that equates to ~1 million total files that can be cached.

In SAP BI Platform 4.0, there are about 100,000 static files that comprise the static content deployed to Apache, which amount to ~2.3GB of content. There's no doubt that that Apache is capable of caching all of it, but do we really want to allocate an additional 2.3GB of disk space just for web cache? Keep in mind that there is a cost on Apache keeping an index of the cache, and letting it grow too large can adversely affect performance.

A good rule of thumb is to allocate only as much disk space as you would be willing to allocate if it were memory instead. Since the cached files are locally stored, operating system in-memory caching will typically be applied to their access also. So although the files are stored on disk, if they are frequently accessed it is likely the operating system will ensure that they are actually served from memory. As such, making sure there is free memory available, we can further increase the performance in which static files are served by Apache. In this example, we'll use a cache size of 500MB, enabling around 25% of the static files to be cached. This seems like a reasonable price to pay for the increased speed mod_cache provides and still allows Apache to effectively manage the cache.

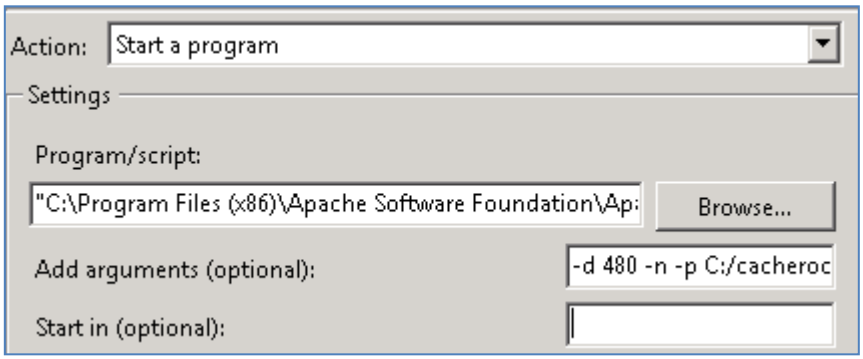
1. Launch the Windows Task Scheduler (Start > Administrative Tools > Task Scheduler)
2. Select 'Create Task' from the Action menu
3. Set the user account of the task to the Apache user, or another user that has Read/Write/Delete privileges to the CacheRoot location and Read/Execute to the Apache directory.



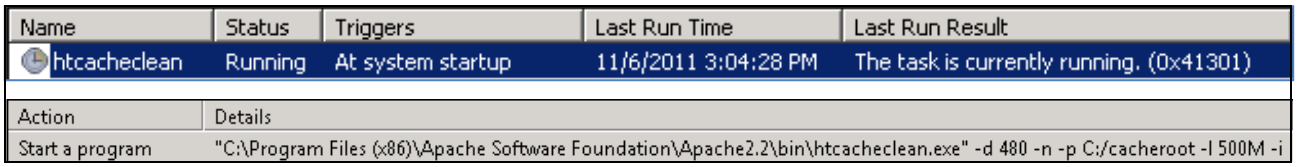
4. We want the task to run even when nobody is logged on to the server so use the radio button highlighted above. The task only needs access to local resources so there is no need to store the password.
5. On the Triggers tab, set the task to begin on startup so that the process is ready to run when Apache starts up.
6. On the Actions tab, browse to htcacheclean at the location `C:\Program Files (x86)\Apache Software Foundation\Apache2.2\bin\htcacheclean.exe`
7. Pass the command line arguments to the process (type this do not copy from the document):

   ```
   -d 480 -n -p C:/cacheroot -l 500M –i
   ```

   The arguments launch the htcacheclean process in unattended mode and have it run every 8 hours (480 minutes). We instruct the process to 'play nice' which means it won't consume large amounts of system resources when running. Point the tool to the CacheRoot directory, set the limit to 500 MB, and be intelligence by only running if there has been a modification of the disk cache.



8. Click 'Ok' to create the task.
9. Right-click the new task and select 'Run'. If it has been configured correctly you should see something like this



Mod_cache and its companions mod_disk_cache and htcacheclean provide powerful caching mechanisms that let us accelerate delivery of BI Launch Pad. When coupled with thorough performance testing the benefits can be measured and used to support the value proposition for using Apache in a deployment of SAP BI Platform 4.0. Please do review the Apache Caching Guide thoroughly before applying any of these settings to a Production environment.

For reference, the final httpd-BI40.conf should look something like this:

```
#====================LoadModules=======================
LoadModule jk_module modules/mod_jk.so
LoadModule headers_module modules/mod_headers.so
LoadModule deflate_module modules/mod_deflate.so
LoadModule cache_module modules/mod_cache.so
LoadModule disk_cache_module modules/mod_disk_cache.so

#====================Configure mod_jk==============
# Where to find workers.properties
```

```
# Update this path to match your conf directory location (put workers.properties next to
httpd.conf)
JkWorkersFile conf/workers.properties
# Where to put jk shared memory
# Write shared memory to the logs directory
JkShmFile logs/mod_jk.shm
# Where to put jk logs
# Update this path to match your logs directory location (put mod_jk.log next to
access_log)
JkLogFile logs/mod_jk.log
# Set the jk log level [debug/error/info]
JkLogLevel    info
# Select the timestamp log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"

#====================Configure mod_headers==============
FileETag None
Header unset ETag

   <FilesMatch "\.(gif|jpe?g|png|html?|js|css)$">
              Header set Cache-Control "public, max-age=315360000"
   </FilesMatch>

#====================Configure mod_deflate==============
AddOutputFilterByType DEFLATE text/html text/xml text/plain text/javascript text/json
application/javascript text/css
#Highest 9 - Lowest 1
DeflateCompressionLevel 9

# Don't compress images
SetEnvIfNoCase Request_URI \
       \.(?:gif|jpe?g|png)$ no-gzip dont-vary

#Optional
#Logging
DeflateFilterNote ratio
LogFormat '"%r" %b (%{ratio}n) "%{User-agent}i"' deflate
CustomLog logs/deflate_log.log deflate

#====================Configure mod_cache==============
<IfModule mod_cache.c>

#Address the Thundering Herd identified in Apache's Caching Guide
CacheLock on
CacheLockPath C:/temp/mod_cache-lock
CacheLockMaxAge 5
#This parameter tells Apache to ignore unique session identifiers when caching
#static content.  SAP BI Platform 4.0 uses the strings jsessionid and bttoken to
#identify user sessions.
CacheIgnoreURLSessionIdentifiers jsessionid bttoken
#Don't cache cookies as they are unique per user
CacheIgnoreHeaders Set-Cookie

#Enable mod_disk_cache instead of mod_mem_cache
<IfModule mod_disk_cache.c>

CacheRoot c:/cacheroot
CacheEnable disk /
CacheDirLevels 2
CacheDirLength 1

</IfModule>
</IfModule>
#====================End httpd-BI40.conf==============
```

Finally, we need to direct Apache to load this file by adding the following line to the very bottom of the httpd.conf file:

```
Include conf/extra/httpd-BI40.conf
```

Save and close the file. Restart Apache to load the properties and enable the tuning options above.

Running WDeploy

The next step in the process is to run the WDeploy tool and split the static and dynamic content. WDeploy also does the necessary work to configure the URI mappings for mod_jk when Apache and Tomcat are located on the same physical server.

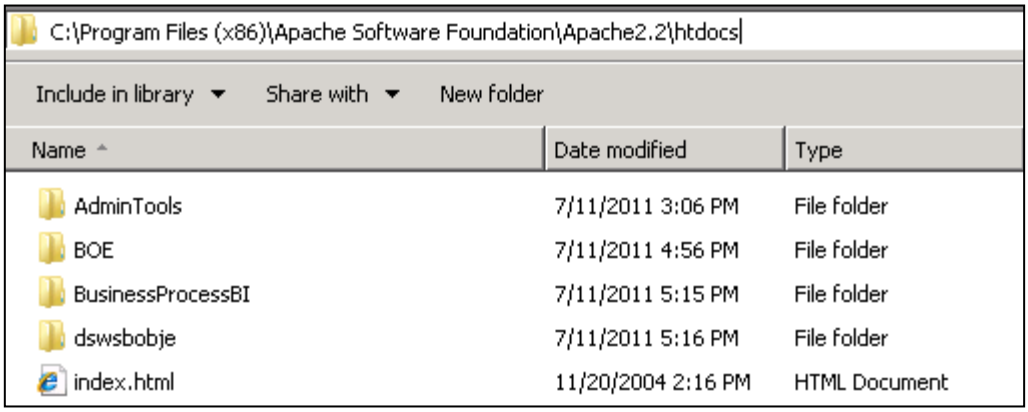
The latest version of WDeploy has a nice GUI tool, in addition to the command line tool, but unfortunately it does not yet support the split deployment option. Therefore we will use the command line version of the tool instead.

1. Launch a command prompt (Start > Run > cmd)
2. `cd C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\wdeploy`
3. Run the command:

   ```
   wdeploy.bat tomcat6 –Das_mode=split –Dws_type=apache deployall
   ```

   This part of the process should divide the static and dynamic content, deploy static content to the Apache ‘htdocs’ directory, and .war files containing dynamic content to Tomcat. WDeploy supports a wide variety of additional deployment options covered in the product documentation.

4. Upon completion, verify the WDeploy process worked properly by looking at the ‘htdocs’ directory in Apache:



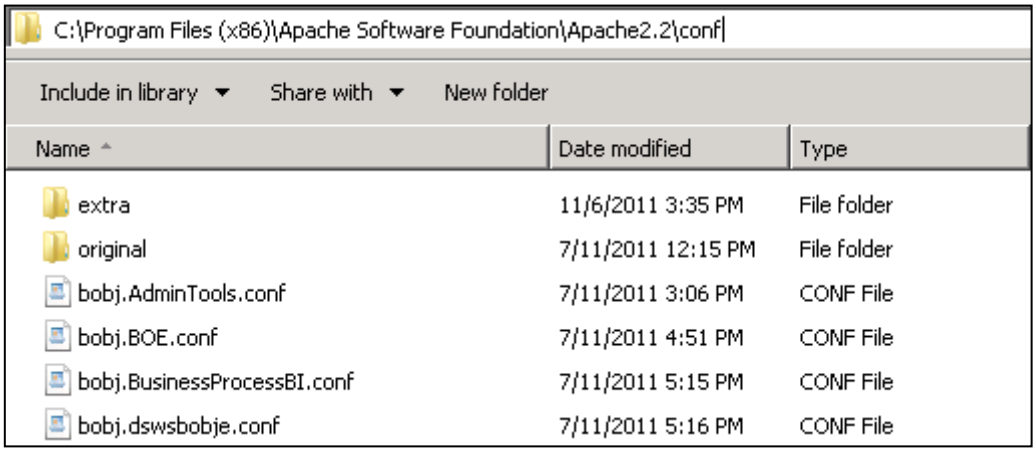
These directories should be populated with all of the static content in SAP BI Platform 4.0. There are approximately 100,000 files in the BOE directory alone.

5. Confirm that mod_jk has been configured correctly by looking for the following lines at the bottom of the httpd.conf file:

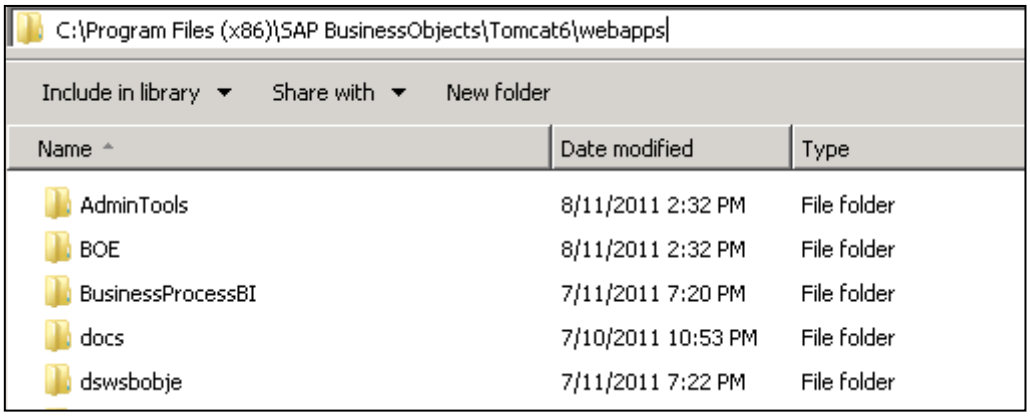
   ```
   Include conf/bobj.AdminTools.conf
   Include conf/bobj.BOE.conf
   ```

```
Include conf/bobj.BusinessProcessBI.conf
Include conf/bobj.dswsbobje.conf
```

These files contain the JkMount commands for SAP BI Platform 4.0 and should be created automatically in the Apache conf directory:



6. Finally, confirm that the .war files have been deployed to Tomcat by checking the contents of the webapps directory:



7. Restart Apache and Tomcat for good measure and the system should be ready to test.

Testing the Proposal

Now that the environment has been setup and the web applications deployed, it is time to test the solution. You should now be able to access the BI Launch Pad through Apache by using the following URL from the web server machine:

http://localhost/BOE/BI

Assuming you get the logon page for BI Launch Pad you will know that mod_jk is working.

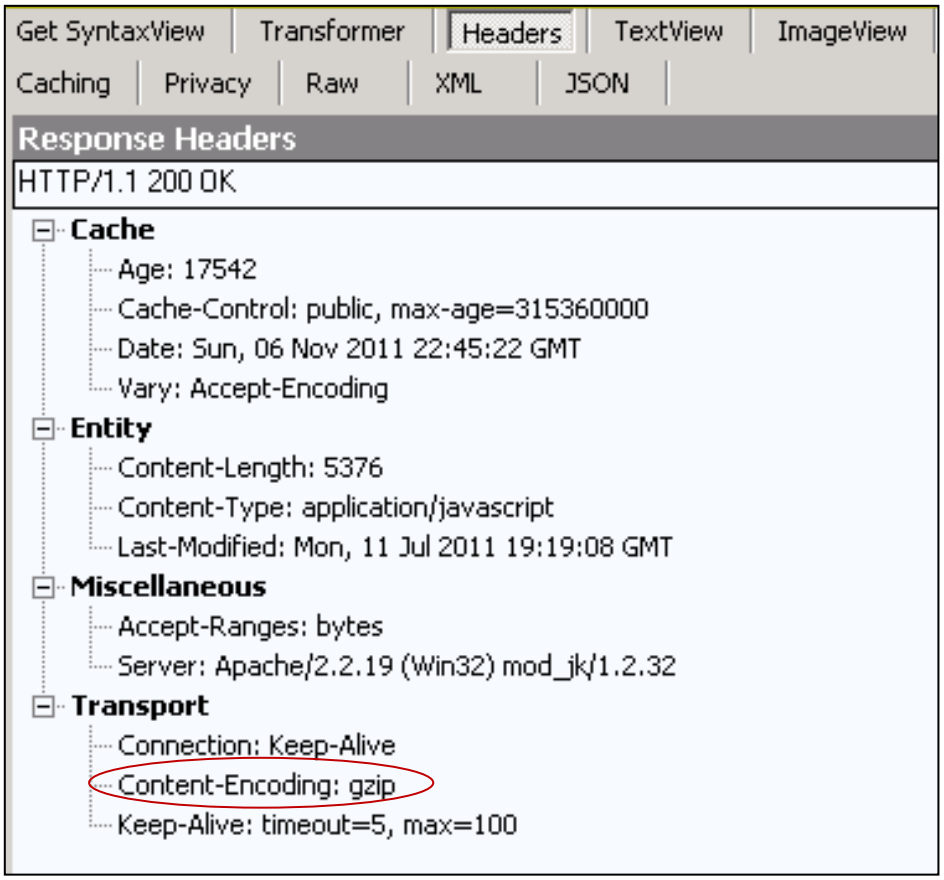
I suggest using an HTTP proxy such as Fiddler to test that the settings we've applied are working properly in your environment. Use Fiddler to record the login process to BI Launch Pad and navigate to the 'Documents' tab.

In Fiddler, select the request utils.js and examine the properties of the object:

URL	Body	Caching
/BOE/BI	220	
/BOE/portal/1107111813/InfoView/logon.jsp	2,356	
/BOE/portal/1107111813/InfoView/common...	3,847	max-age=315360000
/BOE/portal/1107111813/InfoView/js/utils.js	5,376	public, max-age=315360000

You can see that mod_headers is working properly based on the 'public, max-age=315360000' Cache-control header that appears.

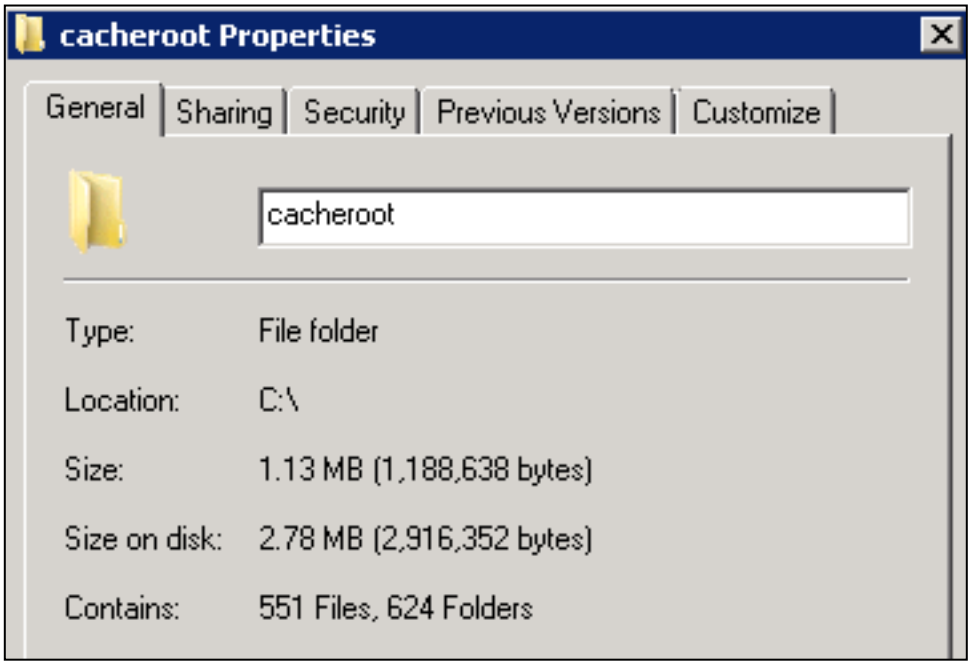
Confirm that mod_deflate is properly compressing the file by clicking the 'Inspectors' tab on the right-hand side, selecting 'Headers' from the bottom half of the pane, and noting the 'Content-Encoding: gzip' header:



You can also tell that the Last-Modified header is in place but that there is no ETag based on the settings we selected in the configuration phase.

Unfortunately Apache does not provide a logging mechanism for mod_cache in Apache 2.2. Logging for mod_cache exists in Apache 2.3 but that release was still in beta at the time this document was written. In lieu of direct logging for mod_cache, the best determination we can make that mod_cache is working as expected is to check the CacheRoot directory and note how many files and folders exist. In an environment

where caching has been primed by a handful of requests, the properties of the CacheRoot directory might look like this:



At a high level this process tells us that each of the components in the proposal are functioning as expected.

Troubleshooting

The sample configuration provided above contains a lot of information and there is a possibility that not everything works as expected initially. A few of the more common problems are highlighted below:

I added the httpd-BI40.conf to my Apache configuration and the Apache service no longer starts…

Typos are an all too common occurrence when adding so much detail to the Apache startup sequence. Fortunately, it is easy to get details on what the problem might be by launching Apache as a process from a command prompt.

1. Launch a command prompt (Start > Run > cmd)
2. Cd C:\Program Files (x86)\Apache Software Foundation\Apache2.2\bin
3. Run the command httpd to launch Apache as a process. You should get some output that helps identify the problem:

```
C:\Program Files (x86)\Apache Software Foundation\Apache2.2\bin>httpd
Syntax error on line 11 of C:/Program Files (x86)/Apache Software Foundation/Apa
che2.2/conf/extra/httpd-BILP.conf:
Invalid command 'JkWorkersFilr', perhaps misspelled or defined by a module not i
ncluded in the server configuration
```

Note that Apache tells us the line number and file where the problem is occurring. In this case, I've simply misspelled the JkWorkersFile command.

I'm not seeing updated Cache-Control headers on my static content…

Try clearing your Internet Explorer browser cache first to ensure fresh copies are requested from the server. If this fails, you may need to delete the contents of the CacheRoot directory. A change to the 'max-age' value in the mod_headers configuration does not cause the static content to be deemed "stale".

I can't browse to BI Launch Pad through Apache. A 404 error, "Page cannot be displayed" is returned…

Mod_jk writes a log file (mod_jk.log) to the `C:\Program Files (x86)\Apache Software Foundation\Apache2.2\logs` directory. This log will tell you about any problems encountered for requests that are supposed to be directed to Tomcat.

For example, if I've forgotten to start my Tomcat server, an error is logged:

```
[error] ajp_send_request::jk_ajp_common.c (1630): (ajp13) connecting to backend failed.
Tomcat is probably not started or is listening on the wrong port (errno=61)
```

This file will usually tell you why you aren't getting to Tomcat as expected. You can also increase the log level of the mod_jk plugin by changing the property `'JkLogLevel'` in `httpd-BI40.conf` to a higher level like debug. Changing this property requires a restart of Apache.

Conclusion

We've now covered a comprehensive proposal for improving the user experience of SAP BI Platform 4.0 by adding an Apache web server to the deployment. By maximizing caching at browser, proxy, and web server levels, we can significantly reduce the amount of time it takes to login and navigate the BI Launch Pad. The concepts covered here should be adapted to fit each environment but, if nothing else, this document should provide a foundation in caching concepts as they apply to SAP BusinessObjects web applications.

In a series of performance tests executed with a default install of SAP BI Platform 4.0, as compared to the configuration described above, average response times were improved by upwards of 25%! Improving the performance of SAP BI Platform 4.0 is a worthwhile exercise and one that every administrator should feel empowered to affect. Ultimately, the user experience in a web application is a telling factor in the overall satisfaction of business users. It should be considered of paramount importance when building and deploying a BI solution.

Related Content

Apache Download Site

Apache Caching Guide

Google Code: Optimizing Cache

Htcacheclean Command Line Reference

For more information, visit the Business Objects homepage

Copyright