Progress®
Kemp® LoadMaster®

Technical Note Caching and Compression

Generated on: 11 March 2025

Copyright

Visit the following page online to see Progress Software Corporation's current Product Documentation Copyright Notice/Trademark Legend: Product Documentation Copyright Notice & Trademarks | Progress

Table of Contents

Chapter 1: Introduction 4

Chapter 2: Caching. 5

Flushing Cache 6

Maximum Cache Size 7

How Caching Works 7

Chapter 3: Data Compression 9

Limit the Maximum Compressible File Size 10

Chapter 4: Caching and Compression UI Options 12

Cache Configuration 12

Compression Options 13

Virtual Service Caching and Compression Options 14

Chapter 5: Caching and Compression Logs 16

Enable Caching and Compression Logs 16

Caching Logs 17

Compression Logs 18

1

Introduction

Introduction

The LoadMaster allows you to cache static content and compress files sent from the LoadMaster.

The LoadMaster advanced caching engine saves valuable Real Server processing power and bandwidth, which can be dedicated to performing critical core business application logic. Significant server performance gains can be achieved when implementing caching. Chatty protocols, such as HTTP, require frequent creating and closing of connections for fetching of static resources, creating unnecessary resource utilization on Real Server(s) and the network.

The LoadMaster data compression feature reduces the amount of data to be transferred for HTTP objects by utilizing gzip compression which is available in all modern web browsers. Leveraging Lempel-Ziv (LZ) compression and HTTP/1.1 GNU zip (gzip) content encoding reduces bandwidth utilization for high compression files such as text files (HTML, CSS, and JavaScript).

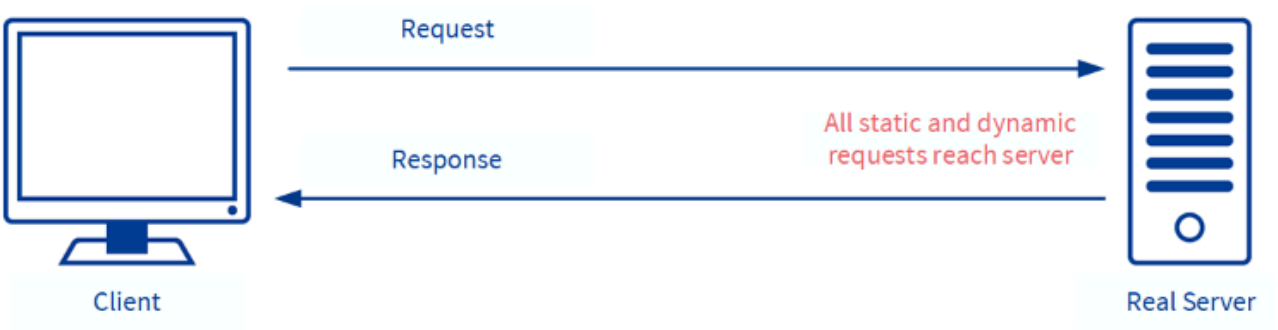
This document provides details about the caching and compression functionality provided by the LoadMaster.

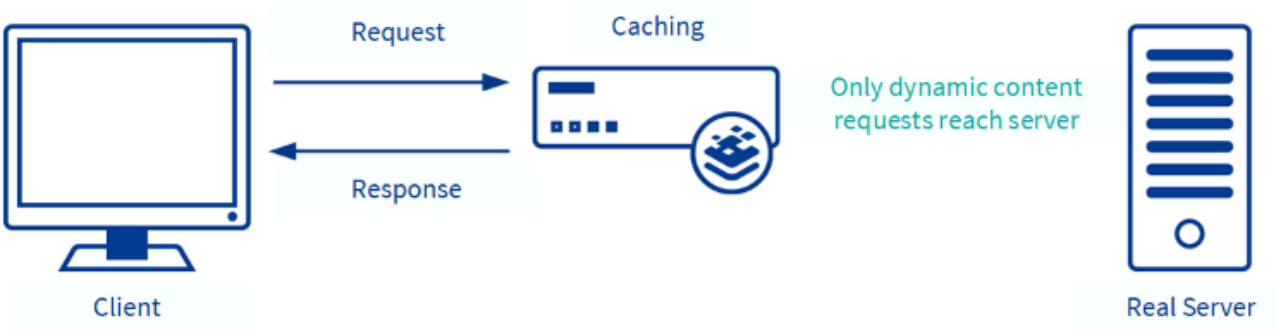
2

Caching

Caching

By deploying LoadMaster caching, your organization can greatly reduce web traffic to the Real Server(s), saving on bandwidth in-front of your Real Server(s).





Caching is available for HTTP and offloaded HTTPS Virtual Services.

Note: HTTP/HTTPS requests with no-cache headers will bypass the cache, following RFC 2616. Cache is filled in a delayed manner - please allow a few seconds for static content to be cached.

Note: In accordance with RFC 2616, URLs which contain query strings (those containing a question mark symbol (**?**) in the rel_path part) will not be cached.

Cache is only stored in memory - it is not stored on disk.

Related Links

- Flushing Cache
- Maximum Cache Size
- How Caching Works

Flushing Cache

Flushing Cache

LoadMaster will not monitor file changes on the Real Server and auto-reload the cache maintained within the Virtual Service. You can force reload the cache by deselecting and selecting the **Enable Caching** checkbox. You can also reload a cached object, sending a non-cache request. Most browsers support this by holding

the left **Shift** key and clicking reload (or pressing **F5**).

Maximum Cache Size

Maximum Cache Size

The amount of global memory (in MB) available for caching can be configured; values have a linear relation to actual memory. To configure this, navigate to **Virtual Services > Cache Configuration**.

The default value for the **Maximum Cache Size** field is **100** MB.

The maximum value for the **Maximum Cache Size** field is 20% of the total memory allocated to the LoadMaster. For example, if the LoadMaster has 4 GB (4,096 MB) of memory, the maximum value of the **Maximum Cache Size** field is **819** MB.

To check the total memory of the LoadMaster, go to **Statistics > Real Time Statistics**.

How Caching Works

How Caching Works

The following steps describe how caching works in the LoadMaster:

1. The LoadMaster checks if the content is already cached.
2. The LoadMaster checks if the status code in the response is valid. The status code must be 200, 203, 300, 301, or 410 for content to be cached.
3. The LoadMaster checks the **Cache-Control** header field for the following directives. If the **Cache-Control** header field has the following data, the LoadMaster will not send the response from the cache:

Note: For further details on the **Cache-Control** header, refer to the following page: Cache-Control.

- **no-cache**
- **no-store**
- **must-revalidate**
- **proxy-revalidate**
- **max-age**
- **Pragma: no-cache**

  Note: For further details on the **Pragma: no-cache** header, refer to the following page: Pragma.

  Note: The **Pragma: no-cache** header is for backwards-compatibility with the HTTP/1.0 caches that do not have a **Cache-Control** HTTP/1.1 header.

1. The **max-age** expiry time is one hour. The LoadMaster checks the response date and last modified time. If there is no last modified time, the date field is used. If the last modified time is less than 60 seconds from the creation date of the file, it will not cache because it has been modified too recently. If the expiry date has passed, the file is not cached.
2. The LoadMaster checks the size of the file to ensure there is enough space to cache.
3. The file is pulled into memory. If there is chunked encoding, it will be de-chunked.
4. If the file should be compressed, it starts compressing.

3

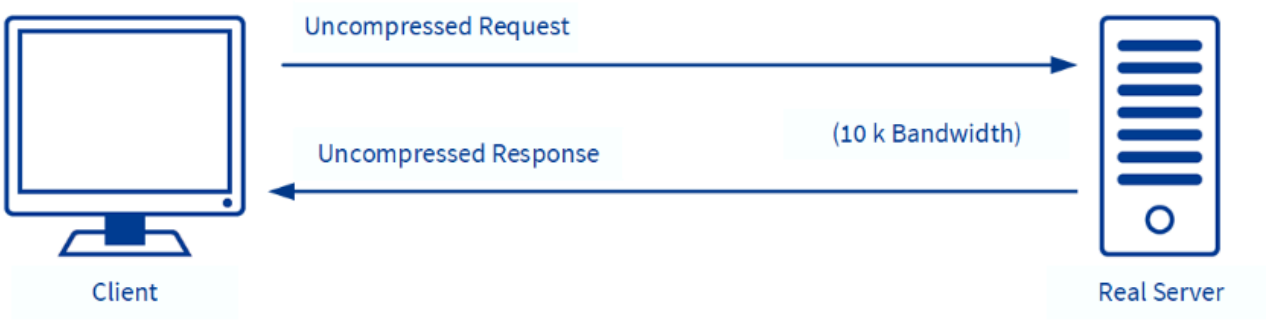
Data Compression

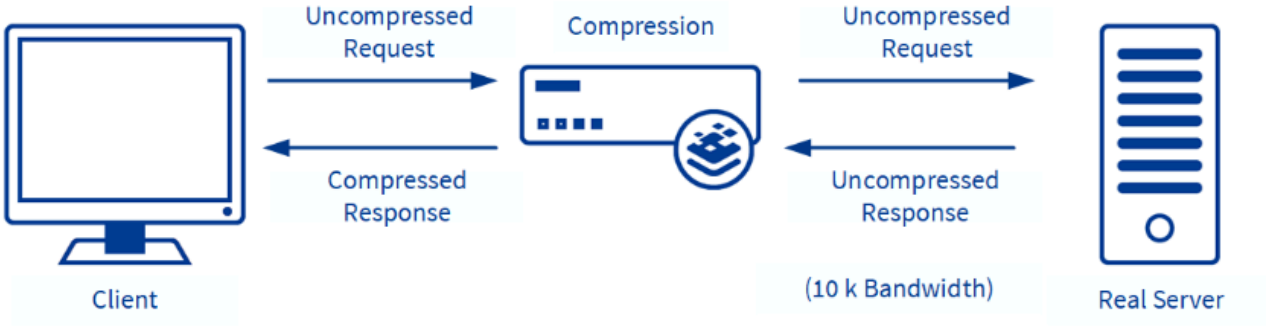
Data Compression

Data compression allows the LoadMaster to compress the application payload per request, reducing network bandwidth consumption without degrading content quality and response time resulting in an improvement for the end-users' overall experience. Data compression is supported on all files. Compression ratios vary by file type.

Note: Compression is not recommended for files 100Mb or greater in size.

Note: Compression only happens once - if the server already compressed the response, the LoadMaster will not compress it again.





Note: The compression feature should be deployed simultaneously with the caching feature to reduce the real-time inline compression requirements. Using only compression can potentially bottleneck the Virtual Service throughput depending on the hardware platform.

Compression can be enabled per HTTP, offloaded HTTPS, and re-encrypted HTTPS Virtual Services.

Compression depends on clients having gzip support. You can verify that a compressed connection to the Virtual Service exists by tracing the client HTTP traffic. If you can locate the **Content-encoding:gzip** header from the LoadMaster your client communication to the LoadMaster is compressed.

Related Links

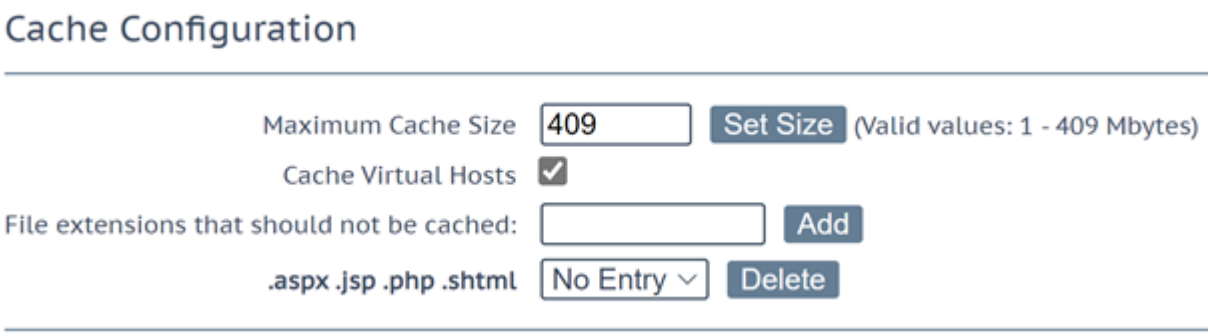
- Limit the Maximum Compressible File Size

Limit the Maximum Compressible File Size

Limit the Maximum Compressible File Size

There are no specific controls in the LoadMaster User Interface (UI) to limit the maximum file size that will be considered for compression when compression is enabled. This can, however, be accomplished by limiting the maximum size of the cache, which determines the maximum file size that can be compressed. This is because any file that is compressed is first cached, so the cache size necessarily limits the maximum compressible file size.

The maximum amount of memory available is 1/5 (one fifth) of the physical memory. The general rule of thumb is that, if there is nothing in the cache, the maximum file size that can be cached and compressed at any given time is equal to 2/3 (two thirds) of the configured maximum cache size minus the amount of space in the cache already allocated to cached files. It allows any files up to 2/3 (two thirds) of the memory. If there are other files in the cache, the limit goes down. Therefore, the actual file size that can be compressed varies according to how much cacheable traffic is flowing through the LoadMaster. Obviously, this can be much lower than the absolute cacheable maximum file size.



This is best illustrated by an example. The default cache size is located under **Virtual Services > Cache Configuration** in the WUI, and is set to 409 Mbytes. Therefore:

1. The amount of space available for the first file that is cached is equal to **409 * 2/3** (since the cache is empty at this time), or about **270 Mbytes**.
2. If a file comes into the cache nanoseconds after the first file, the space available for caching the second file is equal to 270 Mbytes - (minus) [The size of first file remaining in the cache].
3. The reduction of available space in the cache continues as more files are received and enter the cache; space is returned to the available pool as files are compressed and leave the cache, or are aged and removed from the cache as a normal part of cache maintenance.

Careful analysis of the traffic patterns on your LoadMaster is necessary to find the right balance between cache size, compression, and the availability of resources for other resource-intensive operations (such as SSL Offloading and Content Rules).

When a file is rejected for caching or compression, an entry is written in the log that follows this format:

```
Out of memory for cacheing – connection dropped (file size X total size Y)

Out of memory for compressing – connection dropped (file size X total size Y)
```

X is the size of the file that was rejected, and **Y** is the amount of free space in the cache at that time.

Looking for lines like this in the log at varying levels of system performance will help determine how to set the cache size optimally for typical traffic patterns in your environment. To get an accurate picture, this should be done using light to heavy traffic loads with a typical mix of cacheable, compressible, and other resource-intensive traffic that is typical in your configuration.

4

Caching and Compression UI Options

Caching and Compression UI Options

The following sections provide details on the following UI options:

- Global cache configuration and compression options
- Virtual Service-specific caching and compression options

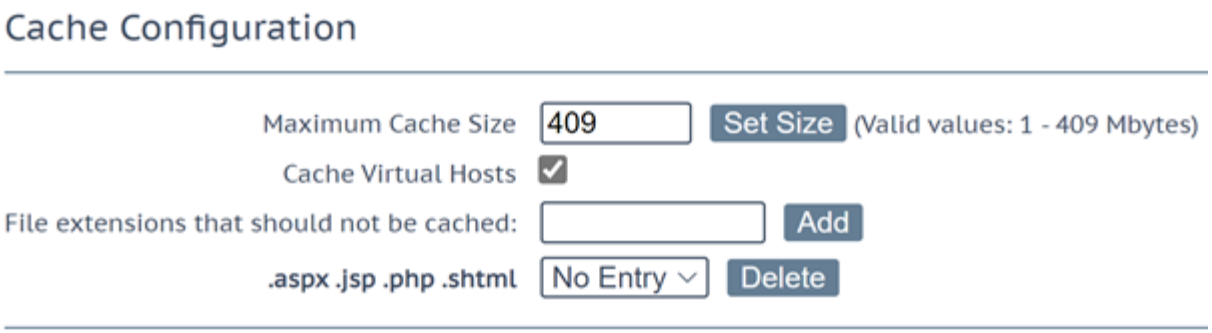
Related Links

- Cache Configuration
- Compression Options
- Virtual Service Caching and Compression Options

Cache Configuration

Cache Configuration

This section provides details on the LoadMaster UI options available in **Virtual Services > Cache Configuration**.



Maximum Cache Size

This defines how much memory can be utilized by the cache in megabytes. The **Maximum Cache Size** defines how much of the main memory should be assigned to the cache. It can never be more than one fifth of the total memory of the machine. Assigning more memory for the cache will reduce the amount of memory available for connections and persist entries. In a system that is correctly configured, there should be enough memory for a full cache and all connections that the system is expected to handle. If this is not the case, the system could run out of memory.

Cache Virtual Hosts

This option is enabled by default. When this option is disabled, the cache presumes there is only one virtual host supported on the Real Server. Enabling this option allows the cache to support multiple virtual hosts which have different content. If a request is cached for one Virtual Service, it will not be cached a second time if the request is sent to another Virtual Service. This is based on the host header of the request.

File extensions that should not be cached

The **File extensions that should not be cached** field and the label below it specifies a list of file types that should not be cached.

The default extensions that are not cached are .aspx, .jsp, .php, and .shtml. You can delete these from the "no cache" list by selecting them and clicking **Delete**.

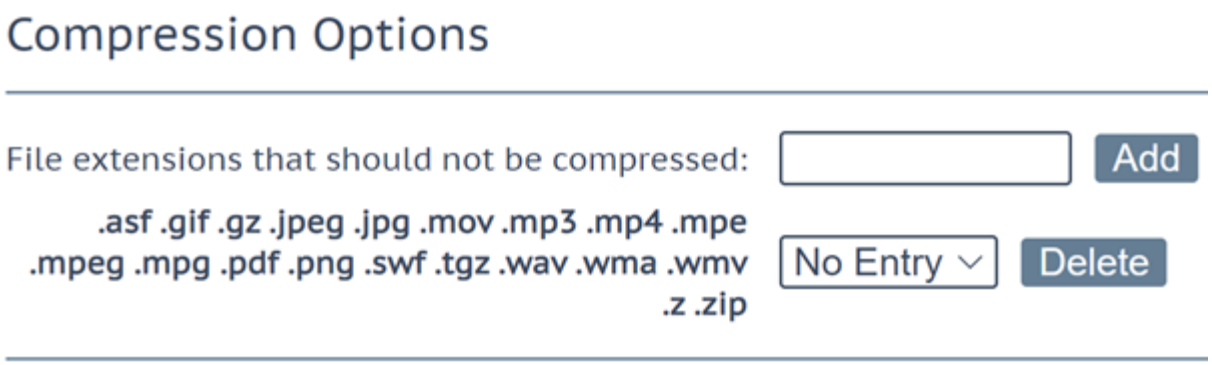
You can add other file extensions to the **File extensions that should not be cached** field.

Items are held in the cache for 15 minutes (or until the cache is flushed by refreshing the page (for example, by pressing **Ctrl + F5** on your keyboard)).

Compression Options

Compression Options

This section provides details on the LoadMaster UI options available in **Virtual Services > Compression Options**.



File extensions that should not be compressed

The **File extensions that should not be compressed** field and the label below it specifies a list of file types that should not be compressed. The default extensions that are not compressed are .asf .gif .gz .jpeg .jpg .mov .mp3 .mp4 .mpe .mpeg .mpg .pdf .png .swf .tgz .wav .wma .wmv .z .zip. You can delete these from the "no compress" list by selecting them and clicking **Delete**.

Virtual Service Caching and Compression Options

Virtual Service Caching and Compression Options

To access the caching and compression options of a Virtual Service, in the LoadMaster UI:

1. Go to **Virtual Services > View/Modify Services**.
2. Click **Modify** on the relevant Virtual Service.
3. Expand the **Advanced Properties** section.

Enable Caching

This option enables caching of static content. This saves valuable Real Server processing power and bandwidth. You can enable caching on HTTP and offloaded HTTPS Virtual Services. You can also enable caching on Virtual Services with SSL re-encryption enabled.

Before caching, the LoadMaster checks if there is any reason the content should not be cached. For example, if it is on the **File extensions that should not be cached** list in **Virtual Services > Cache Configuration**. It also checks if the content is already cached. When the response comes back, the LoadMaster checks if the status code is correct. If the status code is not 200, 203, 300, 301, or 410 - the content is not cached. If the file was created in the last 60 seconds, it will not be cached.

Items are held in the cache for a maximum of 1 hour. This time is not customizable. Every 15 minutes the LoadMaster checks if what is currently in the cache should still be there (it checks the maximum age of the header) and at that point it clears the cache as needed. The cache can be flushed on the browser by refreshing the page (for example, by pressing **Ctrl + F5** on your keyboard). Disabling and re-enabling caching on the Virtual Service flushes the cache for that Virtual Service on the LoadMaster.

Note: Types of files that can be cached may be defined in **Cache Configuration** under the **Virtual Services** menu.

Maximum Cache Usage

If you enable caching on a Virtual Service, you can then limit the amount of cache being used by that service. The LoadMaster reserves 20% of the total system memory for the global cache value. For example, if a LoadMaster has 1GB of RAM - 20% (or 204MB) is reserved for caching and compression.

When you enable caching on a Virtual Service, you then have the option to set **No Limit** (which is the default value) or a percentage usage value. Selecting **No Limit** allows you to use all of the available cache. Alternatively, you can specify a percentage from 1 to 99%. This allows you to break up the 204MB in the example above into a percentage. For example, if you set 20% on the Virtual Service level - the Virtual Service has a total of 40MB available to cache server responses (20% of 204MB = 40.8MB). This leaves approximately 160MB available for other Virtual Services.

Note: This memory is also used by the LoadMaster for content switching and body response rules.

After selecting a percentage, the LoadMaster displays the overall total current usage assigned. This is a combination of the current usage assigned for all Virtual Services. For example, if there are two Virtual Services and on one the **Maximum Cache usage** is set to 10% and on the other it is set to 7%, the **Current usage assigned** value displays 17%. It is possible to over-provision/commit - if you attempt to do this, you will get a message saying **Setting this value will overcommit the Cache memory by N%**.

It is recommended to limit the cache size to prevent unequal use of the cache store. Ensure that the cache maximum usage is adjusted so that each Virtual Service has a percentage of cache to use. If there is no remaining space to be allocated for a cache-enabled Virtual Service, that service will not cache content.

Enable Compression

Files sent from LoadMaster are compressed with Gzip.

Note: If compression is enabled without caching, LoadMaster performance may suffer. When caching and compression are both enabled, caching has priority and therefore the first file is cached but not compressed. On second and subsequent requests, the file is found in the cache and then it is compressed.

The types of files that can be compressed may be defined in **Compression Options** in the **Virtual Services** menu of the LoadMaster UI.

Note: Compression is not recommended for files 100MB or greater in size.

Note: More RAM may need to be added to Virtual LoadMasters using the hypervisor to compress large files.

5

Caching and Compression Logs

Caching and Compression Logs

Refer to the sections below for:

- Instructions on how to enable caching and compression logs
- Some examples of caching and compression logs

Related Links

- Enable Caching and Compression Logs
- Caching Logs
- Compression Logs

Enable Caching and Compression Logs

Enable Caching and Compression Logs

To enable logs relating to caching and compression, follow these steps in the LoadMaster UI:

1. In the main menu, go to **System Configuration > Logging Options > System Log Files**.
2. Click **Debug Options**.
3. Click **Enable Traces**.

4. Click **Back**.
5. Click **View** on the **System Message File** to view the logs.

Caching Logs

Caching Logs

```
2022-05-19T09:46:20+00:00 lb100 kernel: L7: ffff88806f182cd0: Accept on
10.2.122.71:80 from 10.0.37.195:53148 (0)

2022-05-19T09:46:20+00:00 lb100 kernel: L7: ffff88806f182cd0: Parse_http_header

2022-05-19T09:46:20+00:00 lb100 kernel: L7: ffff88806f182cd0: User-Agent:
Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:100.0) Gecko/20100101 Firefox/
100.0

2022-05-19T09:46:20+00:00 lb100 kernel: L7: ffff88806f182cd0:
Parse_http_header: finished

2022-05-19T09:46:20+00:00 lb100 kernel: L7: ffff88806f182cd0: request: GET
/test99.html

2022-05-19T09:46:20+00:00 lb100 kernel: L7: ffff88806f182cd0: RS 10.2.122.8:80
aconns 0 refcnt 2 weight 1000

2022-05-19T09:46:20+00:00 lb100 kernel: L7: Cache-control:
```

Above is an example of L7 debug logs with caching and compression.

If the cache-control header is blank, the LoadMaster can start caching.

If the cache-control header has the following data (or if the Pragma header is set to no-cache) then the LoadMaster will not send a response from the cache:

- no-cache
- no-store
- must-revalidate
- proxy-revalidate
- max-age

```
2022-05-19T09:46:20+00:00 lb100 kernel: L7: In cache flush 0
```

The above line means that a stale cache is being flushed. This happens roughly every 15 minutes. If the line says cache flush 0 it is for a specific Virtual Service. If the line says cache flush 1 is clears all the cache (this is not normally seen in a production setting).

```
2022-05-19T09:46:20+00:00 lb100 kernel: L7: ffff88806f182cd0: cache_found
'/test99.html' 0
```

The above line shows a URL that is being requested - if it is followed by a 0 it means it is a new cache record.

Compression Logs

Compression Logs

```
2022-05-19T10:04:54+00:00 lb100 kernel: L7: ffff88806f157cd0: Connected

2022-05-19T10:04:54+00:00 lb100 kernel: L7: ffff88806f157cd0: updating persist
0 1

2022-05-19T10:04:54+00:00 lb100 kernel: L7: ffff88806f157cd0: Parse_http_header

2022-05-19T10:04:54+00:00 lb100 kernel: L7: ffff88806f157cd0: User-Agent:
Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:100.0) Gecko/20100101 Firefox/
100.0

2022-05-19T10:04:54+00:00 lb100 kernel: L7: ffff88806f157cd0:
Parse_http_header: finished

2022-05-19T10:04:54+00:00 lb100 kernel: L7: ffff88806f157cd0: add_header
Request: GET /test99.html

2022-05-19T10:04:54+00:00 lb100 kernel: L7: ffff88806f157cd0: addadd slen = 533
state = 2 ret = 0

2022-05-19T10:04:54+00:00 lb100 kernel: L7: ffff88806f157cd0: In compress_fill
with

//

2022-05-19T10:04:54+00:00 lb100 kernel: L7: ffff88806f157cd0: ok_to_compress
200
```

The **In compress_fill** with line means that compression is starting.

If the response from the server is HTTP 200, the file can be compressed. If it receives any other response, it cannot be compressed.

```
2022-05-19T10:04:54+00:00 lb100 kernel: L7: ffff88806f157cd0: ok_to_compress
param ffff888070b51118 gzip
```

ffff888070b51118 is an internal pointer in the example above. The LoadMaster checks if the file can be compressed. If the response has already been compressed by the server, the LoadMaster will not compress it a second time. If there is no internal pointer or the value of **content-encoding** is **identity**, this forces the

LoadMaster to compress.