# Configuring Caching and Compression

https://campus.barracuda.com/doc/4259922/

## Caching

Caching stores commonly used information in local memory for quick retrieval to reduce repeated requests to a web server for the same information. Web pages, graphics files, and other objects can be cached, sometimes dramatically improving performance and reliability. Benefits of caching include:

- Reduced latency when retrieving web content.
- An overall reduction in bandwidth and server load.
- Automatic identification and replication of site content.

#### Configure Caching

1. From the **WEBSITES > Traffic Management > Caching** section, identify the service for which you want to enable caching, and click **Edit** next to it.
2. The **Caching** window opens. Specify values for the following fields:
   - **Status** – Set to *On* to enable caching policy for the service.
   - **File Extensions** – Add any file extensions of files you want cached.
   - **Max Size (KB)** – Enter the maximum object size in kilobytes that can be cached.
   - **Min Size (B)** – Enter the minimum object size in bytes that can be cached.
   - **Ignore Request Headers** – If set to *Yes*, HTTP request cache-control headers are ignored for caching decisions (see Object Freshness).
   - **Ignore Response Headers** – If set to *Yes*, HTTP response cache-control headers are ignored for caching decisions (see Object Freshness).
   - **Cache Negative Responses** – If set to *Yes*, HTTP negative responses like 204, 305, 404, 405, 414, 501, 502, and 504 are cached.
   - **Expiry Age (minutes)** – Specify the time in minutes after which the Barracuda Web Application Firewall expires the cached page and goes to the web server for the information requested by the client.
3. Click **Save Changes** to save the above settings.

Objects larger/smaller than the specified Max/Min size will not be cached.

## Content Rules and Dynamic Pages

Requests are compared to web service content rules, and if caching is enabled for the matching rule,

the response is cached and served from the cache for subsequent requests. When disabled, each subsequent request is forwarded to the backend server for the reply.

Caching works well for responses containing static pages which remain unmodified over multiple requests. When response content is likely to change for each request due to context or conditions (for example, when server-side scripting is used to generate context-sensitive responses based on URL/form parameters in requests), a dynamic response is required.

A content rule with a URL of `/reports/*` matches all pages under `/reports`. If cache is enabled on this content rule, all pages under `/reports` are cached. If the `/reports` folder also contains dynamic pages that should not be cached, use one of the following options:

- Move all dynamic pages into another folder, for example `/reports/cgi_bin`, and create a separate content rule for the URL of `/reports/cgi_bin*` disabling caching for this content rule.
- Disable cache for the original content rule, and do not cache any page under `/reports`.

### Object Freshness

The freshness of cached objects determines their life span and whether to retrieve a newer version from the originating server. A freshness algorithm determines when an object is stale and directs corresponding requests to the originating server. Otherwise, the request is served with the cached copy.

The table below provides the algorithm which uses the object's age to determine object freshness. Age is calculated as follows:

```
age = (current_time - time_retrieved) + object_age
```

- When both **Ignore Request Headers** and **Ignore Response Headers** are *Yes*, all objects are considered fresh.
- When **Ignore Request Headers** is *Yes*:
  - If **Ignore Response Headers** is *No* and the age of an object is greater than cached response max-age (if present), the object is considered stale.
- When **Ignore Request Headers** is *No*:
  - If age is greater than request max-age header (if present), the object is considered stale.

The following table describes how to determine an object's freshness.

| If ... | Then object freshness is calculated as ... |
|---|---|
| Ignore Response Headers is *Yes*. | freshness = age - expiry_age |

| | |
|---|---|
| Cached response had an expiration time. | freshness = current_time - object_expiration_time |
| Age of an object is greater than Expiry Age. | freshness = age - expiry_age |
| Cached response has a time last modified header. | stale_age = time_object_retrieved + object_age - time_object_last_modified<br>stale_age = stale_age *age_from_last_modified_percentage / 100)<br>if age > stale_age, freshness = age - stale_age |
| Cached response does not have a time last modified header: | freshness = age - expiry_age |
| Staleness < 0, a min-fresh header request is present, and it is set to be greater than the staleness value (positive value of it). | The object is considered stale. |
| Staleness < 0, and a min-fresh header request is not present. | The object is considered fresh. |
| A max-stale header request is present, and it is set to be greater than the staleness. | The object will expire. |
| A max-stale header request is not present. | The object is considered stale. |

## Compression

Compression improves response time for clients by reducing the quantity of data transferred. Web pages that use HTML, JavaScript, Java, and other text-based languages, can be compressed to improve traffic management and significantly reduce download time. Compression can be applied for all client requests, and to specific client requests that match content rules. Enabling compression for a service applies compression to all service requests.

#### Configure Compression

1. From the **WEBSITES > Traffic Management > Compression** section, identify the service for which you want compression enabled.
2. Click **Edit** next to that service. The **Compression** window appears. Specify values for the following fields:
   - **Status** – Set to *On* to enable compression policy for the service.
   - **Content Types** – Specify the content type(s) to be compressed.
   - **Min Size (B)** – Specify the minimum size for the response.
   - **Compress Unknown Content Types** – Set to Yes to compress files of unknown content type. These unknown file types can be non-text content types such as executable binaries, flash content, and so on.

3. Click **Save Changes**.

- Compression should be turned off for backend web servers. The Barracuda Web Application Firewall will not inspect compressed responses originating from the backend servers. Instead, backend servers should send uncompressed content to the Barracuda Web Application Firewall, which can compress content after applying configured security settings.
- The cache should be cleared when you enable compression to avoid retrieving uncompressed cached objects.