Example Apache .htaccess Configuration

Apache .htaccess files allow users to configure directories of the web server they control without modifying the main configuration file.

While this is useful it's important to note that using `.htaccess` files slows down Apache, so, if you have access to the main server configuration file (which is usually called `httpd.conf`), you should add this logic there under a `Directory` block.

See .htaccess in the Apache HTTPD documentation site for more details about what .htaccess files can do.

The remainder of this document will discuss different configuration options you can add to `.htaccess` and what they do.

Most of the following blocks use the IfModule directive to only execute the instructions inside the block if the corresponding module was properly configured and the server loaded it. This way we save our server from crashing if the module wasn't loaded.

Redirects

There are times when we need to tell users that a resource has moved, either temporarily or permanently. This is what we use `Redirect` and `Redirectmatch` for.

```
# Redirect to a URL on a different host
Redirect "/service" "http://foo2.example.com/service"

# Redirect to a URL on the same host
Redirect "/one" "/two"

# Equivalent redirect to URL on the same host
Redirect temp "/one" "/two"

# Permanent redirect to a URL on the same host
```

```
  Redirect permanent "/three" "/four"

  # Redirect to an external URL
  # Using regular expressions and RedirectMatch
  RedirectMatch "^/oldfile\.html/?$" "http://example.com/
newfile.php"
```

The possible values for the first parameter are listed below. If the first parameter is not included is defaults to `temp`.

permanent
: Returns a permanent redirect status (301) indicating that the resource has moved permanently.

temp
: Returns a temporary redirect status (302). **This is the default**.

seeother
: Returns a "See Other" status (303) indicating that the resource has been replaced.

gone
: Returns a "Gone" status (410) indicating that the resource has been permanently removed. When this status is used the *URL* argument should be omitted.

Cross-origin resources

The first set of directives control CORS (Cross-Origin Resource Sharing) access to resources from the server. CORS is an HTTP-header based mechanism that allows a server to indicate the external origins (domain, protocol, or port) which a browser should permit loading of resources.

For security reasons, browsers restrict cross-origin HTTP requests initiated from scripts. For example, XMLHttpRequest and the Fetch API follow the same-origin policy. A web application using those APIs can only request resources from the same origin the application was loaded from unless the response from other origins includes the apprpriate CORS headers.

General CORS access

This directive will add the CORS header for all resources in the directory from any website.

```
  Header set Access-Control-Allow-Origin "*"
```

Unless you override the directive later in the configuraation or in the configuration of a directory below where you set this one, every request from external servers will be honored, which is unlikely to be what you want.

One alternative is to explicitly state what domains have access to the content of your site. In the example below we restrict access to a subdomain of our main site (example.com). This is more secure and, likely, what you intended to do.

```
  Header set Access-Control-Allow-Origin
"subdomain.example.com"
```

Cross-origin images

As reported in the Chromium Blog and documented in Allowing cross-origin use of images and canvas can lead to fingerprinting attacks.

To mitigate the possibility of these attacks, you should use the `crossorigin` attribute in the images you request and the code snippet below in your `.htaccess` to set the CORS header from the server.

```
     SetEnvIf Origin ":" IS_CORS
     Header set Access-Control-Allow-Origin "*"
env=*IS_CORS*
```

Cross-origin web fonts

Google Chrome's Google Fonts troubleshooting guide tells us that, while Google Fonts may send the CORS header with every response, some proxy servers may strip it before the browser can use it to render the font.

```
    Header set Access-Control-Allow-Origin "*"
```

Cross-origin resource timing

The Resource Timing Level 1 specification defines an interface for web applications to access the complete timing information for resources in a document.

The Timing-Allow-Origin response header specifies origins that are allowed to see values of attributes retrieved via features of the Resource Timing API, which would otherwise be reported as zero due to cross-origin restrictions.

If a resource isn't served with a `Timing-Allow-Origin` or if the header does not include the origin making the request some of the attributes of the `PerformanceResourceTiming` object will be set to zero.

```
    Header set Timing-Allow-Origin: "*"
```

Custom Error Pages/Messsages

Apache allows you to provide custom error pages for users depending on the type of error they receive.

The error pages are presented as URLs. These URLs can begin with a slash (/) for local web-paths (relative to the DocumentRoot), or be a full URL which the client can resolve.

See the ErrorDocument Directive documentation on the HTTPD

documentation site for more information.

```
ErrorDocument 500 /errors/500.html
ErrorDocument 404 /errors/400.html
ErrorDocument 401 https://example.com/subscription_info.html
ErrorDocument 403 "Sorry, can't allow you access today"
```

Error prevention

This setting affects how MultiViews work for the directory the configuration applies to.

The effect of `MultiViews` is as follows: if the server receives a request for /some/dir/foo, if /some/dir has `MultiViews` enabled, and /some/dir/foo does not exist, then the server reads the directory looking for files named foo.*, and effectively fakes up a type map which names all those files, assigning them the same media types and content-encodings it would have if the client had asked for one of them by name. It then chooses the best match to the client's requirements.

The setting disables `MultiViews` for the directory this configuration applies to and prevents Apache from returning a 404 error as the result of a rewrite when the directory with the same name does not exist

```
Options -MultiViews
```

Media Types and Character Encodings

Apache uses mod_mime to assign content metadata to the content selected for an HTTP response by mapping patterns in the URI or filenames to the metadata values.

For example, the filename extensions of content files often define the content's Internet media type, language, character set, and content-encoding. This information is sent in HTTP messages containing that content and used in content negotiation when selecting alternatives, such that the user's preferences are

respected when choosing one of several possible contents to serve.

Changing the metadata for a file does not change the value of the Last-Modified header. Thus, previously cached copies may still be used by a client or proxy, with the previous headers. If you change the metadata (language, content type, character set or encoding) you may need to 'touch' affected files (updating their last modified date) to ensure that all visitors are receive the corrected content headers.

Serve resources with the proper media types (a.k.a MIME types)

Associates media types with one or more extensions to make sure the resources will be served appropriately.

Servers should use text/javascript for JavaScript resourcesn as indicated in the HTML specification

```
  # Data interchange
    AddType application/atom+xml      atom
    AddType application/json          json map topojson
    AddType application/ld+json       jsonld
    AddType application/rss+xml       rss
    AddType application/geo+json      geojson
    AddType application/rdf+xml       rdf
    AddType application/xml           xml
  # JavaScript
    AddType text/javascript           js mjs
  # Manifest files
    AddType application/manifest+json     webmanifest
    AddType application/x-web-app-manifest+json
webapp
    AddType text/cache-manifest           appcache
  # Media files
    AddType audio/mp4                     f4a f4b m4a
    AddType audio/ogg                     oga ogg opus
    AddType image/bmp                     bmp
    AddType image/svg+xml                 svg svgz
    AddType image/webp                    webp
    AddType video/mp4                     f4v f4p m4v mp4
```

```
  AddType video/ogg                     ogv
  AddType video/webm                    webm
  AddType image/x-icon    cur ico
# HEIF Images
  AddType image/heic                    heic
  AddType image/heif                    heif
# HEIF Image Sequence
  AddType image/heics                   heics
  AddType image/heifs                   heifs
# AVIF Images
  AddType image/avif                    avif
# AVIF Image Seuqence
  AddType image/avis                    avis
# WebAssembly
  AddType application/wasm              wasm
# Web fonts
  AddType font/woff                         woff
  AddType font/woff2                        woff2
  AddType application/vnd.ms-fontobject                 eot
  AddType font/ttf                          ttf
  AddType font/collection                   ttc
  AddType font/otf                          otf
# Other
  AddType application/octet-stream          safariextz
  AddType application/x-bb-appworld         bbaw
  AddType application/x-chrome-extension    crx
  AddType application/x-opera-extension     oex
  AddType application/x-xpinstall           xpi
  AddType text/calendar                     ics
  AddType text/markdown                     markdown md
  AddType text/vcard                        vcard vcf
  AddType text/vnd.rim.location.xloc        xloc
  AddType text/vtt                          vtt
  AddType text/x-component                  htc
```

Set the default Charset attribute

Every piece of content on the web has a character set. Most, if not all, the content is UTF-8 Unicode.

Use AddDefaultCharset to serve all resources labeled as `text/html` or `text/plain` with the `UTF-8` charset.

```
AddDefaultCharset utf-8
```

Set the charset for specific media types

Serve the following file types with the `charset` parameter set to `UTF-8` using the AddCharset directive available in `mod_mime`.

```
AddCharset utf-8 .appcache \
  .bbaw \
  .css \
  .htc \
  .ics \
  .js \
  .json \
  .manifest \
  .map \
  .markdown \
  .md \
  .mjs \
  .topojson \
  .vtt \
  .vcard \
  .vcf \
  .webmanifest \
  .xloc
```

Mod_rewrite and the RewriteEngine directives

mod_rewrite provides a way to modify incoming URL requests, dynamically, based on regular expression rules. This allows you to map arbitrary URLs onto

your internal URL structure in any way you like.

It supports an unlimited number of rules and an unlimited number of attached rule conditions for each rule to provide a really flexible and powerful URL manipulation mechanism. The URL manipulations can depend on various tests: server variables, environment variables, HTTP headers, time stamps, external database lookups, and various other external programs or handlers, can be used to achieve granular URL matching.

Enable mod_rewrite

The basic pattern to enable `mod_rewrite` is a pre-requisite for all other tasks that use.

The required steps are:

1. Turn on the rewrite engine (this is necessary in order for the `RewriteRule` directives to work) as documented in the RewriteEngine documentation
2. Enable the `FollowSymLinks` option if it isn't already. See Core Options documentation
3. If your web host doesn't allow the `FollowSymlinks` option, you need to comment it out or remove it, and then uncomment the `Options +SymLinksIfOwnerMatch` line, but be aware of the performance impact
    1. Some cloud hosting services will require you set `RewriteBase`
    2. See Rackspace FAQ and the HTTPD documentation
    3. Depending on how your server is set up, you may also need to use the `RewriteOptions` directive to enable some options for the rewrite engine

```
RewriteEngine On
Options +FollowSymlinks
# Options +SymLinksIfOwnerMatch
# RewriteBase /
# RewriteOptions
```

Forcing https

This Rewrite rules will redirect from the `http://` insecure version to the `https://` secure version of the URL as described in the Apache HTTPD wiki.

```
  RewriteEngine On
  RewriteCond %{HTTPS} !=on

  RewriteRule ^/?(.*) https://%{SERVER_NAME}/$1 [R,L]
```

If you're using cPanel AutoSSL or the Let's Encrypt webroot method to create your SSL certificates, it will fail to validate the certificate if validation requests are redirected to HTTPS. Turn on the condition(s) you need.

```
  RewriteEngine On
  RewriteCond %{HTTPS} !=on
  RewriteCond %{REQUEST_URI} !^/\.well-known/acme-challenge/
  RewriteCond %{REQUEST_URI} !^/\.well-known/cpanel-dcv/[\w-
]+$
  RewriteCond %{REQUEST_URI} !^/\.well-known/pki-
validation/[A-F0-9]{32}\.txt(?:\ Comodo\ DCV)?$
  RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [R=301,L]
```

Redirecting from www. URLs

These directives will rewrite `www.example.com` to `example.com`.

You should not duplicate content in multiple origins (with and without www);This can cause SEO problems (duplicate content), and therefore, you should choose one of the alternatives and redirect the other one. You should also use Canonical URLs to indicate which URL should search engines crawl (if they support the feature).

Set `%{ENV:PROTO}` variable, to allow rewrites to redirect with the appropriate schema automatically (http or https).

The rule assumes by default that both HTTP and HTTPS environments are available for redirection.

```
    RewriteEngine On
```

```
  RewriteCond %{HTTPS} =on
  RewriteRule ^ - [E=PROTO:https]
  RewriteCond %{HTTPS} !=on
  RewriteRule ^ - [E=PROTO:http]

  RewriteCond %{HTTP_HOST} ^www\.(.+)$ [NC]
  RewriteRule ^ %{ENV:PROTO}://%1%{REQUEST_URI} [R=301,L]
```

Inserting the www. at the beginning of URLs

These rules will insert `www.` at the beginning of a URL. It's important to note that you should never make the same content available under two different URLs.

This can cause SEO problems (duplicate content), and therefore, you should choose one of the alternatives and redirect the other one. For search engines that support them you should use Canonical URLs to indicate which URL should search engines crawl.

Set `%{ENV:PROTO}` variable, to allow rewrites to redirect with the appropriate schema automatically (http or https).

The rule assumes by default that both HTTP and HTTPS environments are available for redirection. If your SSL certificate could not handle one of the domains used during redirection, you should turn the condition on.

The following might not be a good idea if you use "real" subdomains for certain parts of your website.

```
RewriteEngine On
RewriteCond %{HTTPS} =on
RewriteRule ^ - [E=PROTO:https]
RewriteCond %{HTTPS} !=on
RewriteRule ^ - [E=PROTO:http]

RewriteCond %{HTTPS} !=on

RewriteCond %{HTTP_HOST} !^www\. [NC]
RewriteCond %{SERVER_ADDR} !=127.0.0.1
RewriteCond %{SERVER_ADDR} !=::1
```

```
  RewriteRule ^
%{ENV:PROTO}://www.%{HTTP_HOST}%{REQUEST_URI} [R=301,L]
```

Frame Options

The example below sends the `X-Frame-Options` response header with the value `DENY`, informing browsers not to display the content of the web page in any frame to protect website against clickjacking.

This might not be the best setting for everyone. You should read about the other two possible values for the `X-Frame-Options` header: `SAMEORIGIN` and `ALLOW-FROM`.

While you could send the `X-Frame-Options` header for all of your website's pages, this has the potential downside that it forbids even any framing of your content (e.g.: when users visit your website using a Google Image Search results page).

Nonetheless, you should ensure that you send the `X-Frame-Options` header for all pages that allow a user to make a state-changing operation (e.g: pages that contain one-click purchase links, checkout or bank-transfer confirmation pages, pages that make permanent configuration changes, etc.).

```
  Header always set X-Frame-Options "DENY"
"expr=%{CONTENT_TYPE} =~ m#text/html#i"
```

Content Security Policy (CSP)

CSP (Content Security Policy) mitigates the risk of cross-site scripting and other content-injection attacks by setting a `Content Security Policy` which whitelists trusted sources of content for your website.

There is no policy that fits all websites, the example below is meant as guidelines for you to modify for your site.

The example policy below:

1. Restricts all fetches by default to the origin of the current website by setting the `default-src` directive to `'self'` - which acts as a fallback to all Fetch directives.
    1. This is convenient as you do not have to specify all Fetch directives that apply to your site, for example: `connect-src 'self'; font-src 'self'; script-src 'self'; style-src 'self'`, etc
    2. This restriction also means that you must explicitly define from which site(s) your website is allowed to load resources from, otherwise it will be restricted to the same origin as the page making the request
2. Disallows the <base> element on the website. This is to prevent attackers from changing the locations of resources loaded from relative URLs
    1. If you want to use the <base> element, then use `base-uri 'self'` instead
2. Only allows form submissions are from the current origin with: `form-action 'self'`
3. Prevents all websites (including your own) from embedding your webpages within e.g. the <iframe> or <object> element by setting: `frame-ancestors 'none'`.
    1. The `frame-ancestors`directive helps avoid "Clickjacking" attacks and is similar to the `X-Frame-Options` header
    2. Browsers that support the CSP header will ignore `X-Frame-Options` if `frame-ancestors` is also specified
4. Forces the browser to treat all the resources that are served over HTTP as if they were loaded securely over HTTPS by setting the `upgrade-insecure-requests` directive
    1. `upgrade-insecure-requests` **does not ensure HTTPS for the top-level navigation. If you want to force the website itself to be loaded over HTTPS you must include the** `Strict-Transport-Security` **header**
2. Includes the `Content-Security-Policy` header in all responses that are able to execute scripting. This includes the commonly used file types: HTML, XML and PDF documents. Although Javascript files can not execute scripts in a "browsing context", they are included to target web workers

    To make your CSP implementation easier, you can use an online CSP header generator. You should also use a validator to make sure your header does what you want it to do.

```
  Content-Security-Policy "default-src 'self'; base-uri
'none'; form-action 'self'; frame-ancestors 'none';
upgrade-insecure-requests" "expr=%{CONTENT_TYPE} =~
m#text\/(html|javascript)|application\/pdf|xml#i"
```

Directory access

This directive will prevent access to directories that don't have an index file present in whatever format the server is configured to use, like `index.html`, or `index.php`.

```
    Options -Indexes
```

Block access to hidden files and directories

In Macintosh and Linux systems, files that begin with a period are hidden from view but not from access if you know their name and location. These types of files usually contain user preferences or the preserved state of a utility, and can include rather private places like, for example, the `.git` or `.svn` directories.

The `.well-known/` directory represents the standard (RFC 5785) path prefix for "well-known locations" (e.g.: `/.well-known/manifest.json`, `/.well-known/keybase.txt`), and therefore, access to its visible content should not be blocked.

```
    RewriteEngine On
    RewriteCond %{REQUEST_URI} "!(^|/)\.well-
known/([^./]+./?)+$" [NC]
    RewriteCond %{SCRIPT_FILENAME} -d [OR]
    RewriteCond %{SCRIPT_FILENAME} -f
    RewriteRule "(^|/)\." - [F]
```

Block access to files with sensitive information

Block access to backup and source files that may be left by some text editors and can pose a security risk when anyone has access to them.

Update the `<FilesMatch>` regular expression in the following example to include any files that might end up on your production server and can expose sensitive information about your website. These files may include: configuration files or files that contain metadata about the project among others.

```
        Require all denied
```

HTTP Strict Transport Security (HSTS)

If a user types `example.com` in their browser, even if the server redirects them to the secure version of the website, that still leaves a window of opportunity (the initial HTTP connection) for an attacker to downgrade or redirect the request.

The following header ensures that a browser only connects to your server via HTTPS, regardless of what the users type in the browser's address bar.

Be aware that Strict Transport Security is not revokable and you must ensure being able to serve the site over HTTPS for as long as you've specified in the `max-age` directive. If you don't have a valid TLS connection anymore (e.g. due to an expired TLS certificate) your visitors will see an error message even when attempting to connect over HTTP.

```
  # Header always set
  Strict-Transport-Security "max-age=16070400;
includeSubDomains" "expr=%{HTTPS} == 'on'"
  # (1) Enable your site for HSTS preload inclusion.
  # Header always set
  Strict-Transport-Security "max-age=31536000;
includeSubDomains; preload" "expr=%{HTTPS} == 'on'"
```

Prevent some browsers from MIME-sniffing the response

Some older browsers would try and guess the content type of a resource, even when it isn't properly set up on the server configuration.

This reduces exposure to drive-by download attacks and cross-origin data leaks.

```
    Header always set X-Content-Type-Options "nosniff"
```

Referrer Policy

We inlcude the `Referrer-Policy` header is included in responses for resources that are able to request (or navigate to) other resources.

This includes commonly used resource types: HTML, CSS, XML/SVG, PDF documents, scripts and workers.

To prevent referrer leakage entirely, specify the `no-referrer` value instead. Note that the effect could impact analytics metrics negatively.

Use services like the ones below to check your Referrer Policy:

1. securityheaders.com
2. Mozilla Observatory

```
  Header always set Referrer-Policy "strict-origin-when-
cross-origin" "expr=%{CONTENT_TYPE} =~
m#text\/(css|html|javascript)|application\/pdf|xml#i"
```

Disable TRACE HTTP Method

The TRACE method, while seemingly harmless, can be successfully leveraged in some scenarios to steal legitimate users' credentials. See A Cross-Site Tracing (XST) attack and OWASP Web Security Testing Guide

Modern browsers now prevent TRACE requests being made via JavaScript, however, other ways of sending TRACE requests with browsers have been discovered, such as using Java.

If you have access to the main server configuration file, use the `TraceEnable` directive instead.

```
RewriteEngine On
RewriteCond %{REQUEST_METHOD} ^TRACE [NC]
RewriteRule .* - [R=405,L]
```

Remove the X-Powered-By response header

Some frameworks like PHP and ASP.NET set an `X-Powered-By` header that contains information about them (e.g.: their name, version number)

This header doesn't provide any value, and in some cases, the information it provides can expose vulnerabilities

```
  Header unset X-Powered-By
  Header always unset X-Powered-By
```

If you can, you should disable the `X-Powered-By` header from the language/framework level (e.g.: for PHP, you can do that by setting the folowing in `php.ini`.

```
expose_php = off;
```

Remove Apache-generated Server Information Footer

Prevent Apache from adding a trailing footer line containing information about the server to the server-generated documents (e.g.: error messages, directory listings, etc.).

See ServerSignature Directive for more information on what the server signature provides and the ServerTokens Directive for information about configuring the information provided in the signature.

```
ServerSignature Off
```

Fix broken AcceptEncoding Headers

Some proxies and security software mangle or strip the `Accept-Encoding` HTTP header. See Pushing Beyond Gzipping for a more detailed explanation.

```

```

Compress media types

Compress all output labeled with one of the following media types using the AddOutputFilterByType Directive.

```
AddOutputFilterByType DEFLATE "application/atom+xml" \
  "application/javascript" \
  "application/json" \
  "application/ld+json" \
  "application/manifest+json" \
  "application/rdf+xml" \
  "application/rss+xml" \
  "application/schema+json" \
  "application/geo+json" \
  "application/vnd.ms-fontobject" \
  "application/wasm" \
  "application/x-font-ttf" \
  "application/x-javascript" \
  "application/x-web-app-manifest+json" \
  "application/xhtml+xml" \
  "application/xml" \
  "font/eot" \
  "font/opentype" \
  "font/otf" \
  "font/ttf" \
  "image/bmp" \
  "image/svg+xml" \
  "image/vnd.microsoft.icon" \
  "text/cache-manifest" \
  "text/calendar" \
  "text/css" \
  "text/html" \
  "text/javascript" \
  "text/plain" \
  "text/markdown" \
  "text/vcard" \
  "text/vnd.rim.location.xloc" \
  "text/vtt" \
  "text/x-component" \
  "text/x-cross-domain-policy" \
  "text/xml"
```

Map extensions to media types

Map the following filename extensions to the specified encoding type using AddEncoding so Apache can serve the file types with the appropriate `Content-Encoding` response header (this will NOT make Apache compress them!).

If these files types would be served without an appropriate `Content-Encoding` response header, client applications (e.g.: browsers) wouldn't know that they first need to uncompress the response, and thus, wouldn't be able to understand the content.

```
    AddEncoding gzip              svgz
```

Cache expiration

Serve resources with a far-future expiration date using the mod_expires module, and Cache-Control and Expires headers.

```
    ExpiresActive on
    ExpiresDefault
"access plus 1 month"

  # CSS
    ExpiresByType text/css
"access plus 1 year"
  # Data interchange
    ExpiresByType application/atom+xml
"access plus 1 hour"
    ExpiresByType application/rdf+xml
"access plus 1 hour"
    ExpiresByType application/rss+xml
"access plus 1 hour"
    ExpiresByType application/json
"access plus 0 seconds"
    ExpiresByType application/ld+json
"access plus 0 seconds"
    ExpiresByType application/schema+json
```

```
"access plus 0 seconds"
    ExpiresByType application/geo+json
"access plus 0 seconds"
    ExpiresByType application/xml
"access plus 0 seconds"
    ExpiresByType text/calendar
"access plus 0 seconds"
    ExpiresByType text/xml
"access plus 0 seconds"
  # Favicon (cannot be renamed!) and cursor images
    ExpiresByType image/vnd.microsoft.icon
"access plus 1 week"
    ExpiresByType image/x-icon
"access plus 1 week"
  # HTML
    ExpiresByType text/html
"access plus 0 seconds"
  # JavaScript
    ExpiresByType text/javascript
"access plus 1 year"
  # Manifest files
    ExpiresByType application/manifest+json
"access plus 1 week"
    ExpiresByType application/x-web-app-manifest+json
"access plus 0 seconds"
    ExpiresByType text/cache-manifest
"access plus 0 seconds"
  # Markdown
    ExpiresByType text/markdown
"access plus 0 seconds"
  # Media files
    ExpiresByType audio/ogg
"access plus 1 month"
    ExpiresByType image/bmp
"access plus 1 month"
    ExpiresByType image/gif
"access plus 1 month"
    ExpiresByType image/jpeg
"access plus 1 month"
    ExpiresByType image/svg+xml
"access plus 1 month"
    ExpiresByType image/webp
"access plus 1 month"
```

```
    # PNG and animated PNG
    ExpiresByType image/apng
"access plus 1 month"
    ExpiresByType image/png
"access plus 1 month"
    # HEIF Images
    ExpiresByType image/heic
"access plus 1 month"
    ExpiresByType image/heif
"access plus 1 month"
    # HEIF Image Sequence
    ExpiresByType image/heics
"access plus 1 month"
    ExpiresByType image/heifs
"access plus 1 month"
    # AVIF Images
    ExpiresByType image/avif
"access plus 1 month"
    # AVIF Image Sequence
    ExpiresByType image/avis
"access plus 1 month"
    ExpiresByType video/mp4
"access plus 1 month"
    ExpiresByType video/ogg
"access plus 1 month"
    ExpiresByType video/webm
"access plus 1 month"


  # WebAssembly

    ExpiresByType application/wasm
"access plus 1 year"


  # Web fonts

    # Collection
    ExpiresByType font/collection
"access plus 1 month"

    # Embedded OpenType (EOT)
    ExpiresByType application/vnd.ms-fontobject
```

```
"access plus 1 month"
    ExpiresByType font/eot
"access plus 1 month"

    # OpenType
    ExpiresByType font/opentype
"access plus 1 month"
    ExpiresByType font/otf
"access plus 1 month"

    # TrueType
    ExpiresByType application/x-font-ttf
"access plus 1 month"
    ExpiresByType font/ttf
"access plus 1 month"

    # Web Open Font Format (WOFF) 1.0
    ExpiresByType application/font-woff
"access plus 1 month"
    ExpiresByType application/x-font-woff
"access plus 1 month"
    ExpiresByType font/woff
"access plus 1 month"

    # Web Open Font Format (WOFF) 2.0
    ExpiresByType application/font-woff2
"access plus 1 month"
    ExpiresByType font/woff2
"access plus 1 month"
  # Other

    ExpiresByType text/x-cross-domain-policy
"access plus 1 week"
```