Static cache

The static cache for hosts: pages generated as plain files and served by the web server without PHP, why it ships off and the installation book says to leave it off, when a customer might still want it, how to generate…

Source file: `doc/eng-us/help/hosting/performance/static-cache.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/hosting/performance/static-cache.md

This page is for system administrators at hosting companies who are asked about the static cache, or who see it mentioned in old hosting guides. It answers: what it is, why it ships off, when it still makes sense, and how to generate and serve it if a customer needs it. The sources are Static cache generator, Static cache defaults and Chapter 10, section 10.12.

> **Warning**
>
> `.htaccess` **slows down every request.** When `.htaccess` files are enabled (`AllowOverride` anything but `None`), let alone used, Apache looks for them and reads and evaluates them on every request, in every directory of the path, and page loading times go up drastically (measured). The only real fix is to switch the feature off for the virtual host: `AllowOverride None` in the HTTPS virtual host, and the rules in the virtual host's own configuration as mod_rewrite rules. That is far faster in production speed tests and well worth the time it takes to optimize a production configuration. Switch over right away: copy the rules of the example HTTPS virtual host into your virtual host (with the static cache, also `staticcache_offload.conf`), set `AllowOverride None`, run `apachectl configtest`, reload Apache and test the site. The same rules for other servers: nginx, LiteSpeed, Caddy, IIS, Velocity (built in); where virtual host rules go in a control panel: Plesk, cPanel, DirectAdmin, ISPConfig. **On LiteSpeed or OpenLiteSpeed,** the server's own page cache can also answer anonymous visitors and crawlers without PHP: Page caching with LiteSpeed and OpenLiteSpeed.

Contents

1. What it is
2. Why it ships off
3. When a customer might still want it
4. Generating it
5. Serving it from the web server
6. Settings
7. Checklist

What it is

The static cache turns public pages into plain files under `var/<site>/static` that the web server sends without starting Exponential at all. Rewrite rules answer a request from the file when it is a plain `GET` or `HEAD`, has no query string and carries no session; forms, searches and anyone signed in always get the live page.

Why it ships off

`site.ini [ContentSettings] StaticCache=disabled` is the shipped value, and the installer writes it so. The installation book is direct: "**Leave the static cache off.** ... With the static cache on, every publish regenerated every affected page on every public siteaccess over HTTP inside the editor's request, so a publish that changed nothing took seconds instead of a fraction of one. Velocity's response cache gives the same benefit to visitors without that cost." (Chapter 10, section 10.12)

The 6.0 work made the refresh cheaper (each page fetched once, several at a time: a no-change publish that refreshed 24 pages in 4975 ms now refreshes 14 pages in about 0.5 s), but the cost still lands in the editor's request (Static cache generator).

	Static cache	Velocity response cache	Role-aware HTTP cache
Ships	off	on under Velocity	off
Served by	the web server, from a file	Velocity's parent process	an early exit or Velocity, before the kernel
Signed-in visitors	never	never	yes, per permission context
Cost on publish	pages refreshed over HTTP in the editor's request	none; entries expire	purges, no refetch

When a customer might still want it

Where neither Velocity nor the HTTP cache is available and a site is almost entirely public pages that rarely change (the feature page's own description: "sites with mostly public, rarely changing pages"). Agree the publish cost with the customer first. With `[CacheSettings] CronjobCacheClear=enabled` the `staticcache_cleanup` cronjob part does the clearing instead of the publish.

Generating it

From **Setup > Cache > Static content cache**: choose a public site, a page limit (2500, 1 to 20000) and a link depth (12, 0 to 30), press *Create new* and watch each page being stored. From the command line:

Shell

```
php bin/php/makestaticcache.php --help --allow-root-user
php bin/php/makestaticcache.php --site=site --allow-root-user
php bin/php/makestaticcache.php --site=site --max-pages=500 --max-depth=6 --allow-root-user
```

The generator crawls the site from its own root and follows only the links it writes, so it visits exactly the addresses a visitor can request. A run empties the site's stored pages first unless `--keep` is given. Run it as the site's user so the files stay writable for the web server.

Serving it from the web server

Setup	What to use
Apache with `.htaccess`	copy `.htaccess_root_static` to `.htaccess` instead of `.htaccess_root`; it is `.htaccess_root` plus the offload rules. From the console: `./console exp:setup:htaccess --static --apply` (add `--force` to replace an existing `.htaccess`; it is backed up first)
Apache virtual host	the rules of doc/eng-us/examples/staticcache_offload.conf in the `<Directory>` section of the **public** site's vhost
other servers	reproduce the same conditions (method, empty query string, no session cookie, file exists)

Check the line `RewriteRule ^ - [E=EZ_VAR_SITE:site]`: the value is `site.ini [FileSettings] VarDir` without the leading `var/`; a default installation (`var/site`) needs no change.

> **Warning**
>
> Never add the offload rules to the administration vhost: "nothing there is cached, and the rules must never be given a chance to answer an editor's request from a file generated for an anonymous visitor" (`doc/eng-us/examples/staticcache_offload.conf`).

Settings

`settings/staticcache.ini`, `[CacheSettings]` (override in `settings/override/staticcache.ini.append.php`):

Key	Default	Meaning
`HostName`	empty	deprecated; leave empty so each siteaccess is fetched from its own `SiteURL`
`SourceProtocol`	`http`	`http` or `https` for the fetch
`StaticStorageDir`	`static`	becomes `var/<site>/static`; an absolute path is used as given
`MaxCacheDepth`	12	deeper URLs are neither cached nor refreshed on publish
`CachedURLArray[]`	`/`, `/*`	URLs to cache
`CachedSiteAccesses[]`	empty	empty means every public siteaccess
`AlwaysUpdateArray[]`	`/`	URLs refreshed on every publish
`CronjobCacheClear`	`disabled`	`enabled` lets the `staticcache_cleanup` cronjob part clear instead of the publish
`FetchConcurrency`	8 (in code)	pages fetched at once on refresh

Installations upgraded from older releases may have overrides that relied on the old defaults; see Static cache defaults.

Checklist

- ☐ `StaticCache=disabled` on every plan unless a customer asked for it and accepted the publish cost.

☐ If used: offload rules only on the public vhost; `EZ_VAR_SITE` matches `VarDir`.

☐ Generation run as the site's user; the stored pages checked with a plain `curl`.

☐ Signed-in requests and requests with a query string still reach PHP.

Related pages

- Caching layers
- Velocity response cache
- Apache with PHP-FPM
- Static cache generator
- Setting up .htaccess and config.php from the console
- Free documentation: this page is free to copy, translate and improve under the GNU FDL 1.3.

> **Become an Exponential Partner**
>
> Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.
>
> Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.