The role-aware HTTP cache

Source file: doc/eng-us/bc/6.0/httpcache.md Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/bc/6.0/httpcache.md

Read this page if you want to serve whole pages from a cache to signed-in visitors as well as anonymous ones, or if you see an `X-Exp-Cache` header and want to know what it means. The role-aware HTTP cache keeps whole rendered pages per permission context and serves them before the kernel starts. It is off by default (`settings/httpcache.ini`, `Enabled=disabled`); nothing changes until you switch it on.

In short

What changed	New cache `httpcache.ini`; early exit `kernel/private/classes/httpcache/ezphttpcacheearlyexit.php`; Velocity serves the same entries from its server process.
Who is affected	Nobody until `Enabled=enabled`. Then: sites with session storage other than files, and templates that print personal data.
How to check	`curl -sI https://your-host/ | grep -i x-exp-cache` shows `HIT`, `STALE`, `MISS (reason)` or `BYPASS (reason)` once it is on.
How to switch on	Follow the three steps of "Switching it on".

What it does

A content view (`content/view`, every URL alias) is stored after it is rendered, keyed by scheme, host, siteaccess, path and the visitor's **permission context**:

- **anonymous** — one context for everyone who is not signed in;
- **role context** — shared by every user with the same roles and the same role assignment limitations, so a thousand members share one copy of each page;
- **private context** — one user alone, when a `content/read` policy depends on who they are (`Owner`, `ParentOwner`, `Group`, `ParentGroup`, `User_Section`, `User_Subtree`).

The next request for the same page in the same context is answered from the store:

- under Apache or php-fpm by `kernel/private/classes/httpcache/ezphttpcacheearlyexit.php`, included from `config.php`, before the autoloader, INI files or database;
- under Exponential Velocity by the server process itself (`Q.web.appCache`, written by `exp:velocity`), before a worker is woken.

Both use the same class, `ezpHttpCacheContract`, so they compute the same keys and serve the same bytes. The visitor's form token is taken out of the stored page and the visitor's own put back in on every hit.

Measured on a reference installation (2026-09-26, one content page, 8 concurrent)

	Hit	Full render
Apache, anonymous	508 req/s, p50 14 ms	28 req/s, p50 220 ms
Apache, signed in	484 req/s, p50 15 ms	4.3 req/s, p50 1.86 s
Velocity, anonymous	1392 req/s, p50 4 ms	—
Velocity, signed in	1156 req/s, p50 5 ms	4.3 req/s, p50 1.84 s

Switching it on

INI

```ini
# settings/override/httpcache.ini.append.php
[HttpCacheSettings]
Enabled=enabled
```

Then:

1. **Include the early exit from** `config.php` (Apache, php-fpm, FrankenPHP):

   PHP

   ```php
   if ( is_file( __DIR__ . '/kernel/private/classes/httpcache/ezphttpcacheearlyexit.php' ) )
       require __DIR__ . '/kernel/private/classes/httpcache/ezphttpcacheearlyexit.php';
   ```

2. **Keep sessions in files that every server can read.** The early exit finds the visitor from the session file named by the session cookie; it cannot ask Redis or a database. Every server that serves a cached siteaccess must use the same directory — Apache's pool and Velocity (`[PHPSettings] IniOptions[]=session.save_path=0;0660;<dir>` in `velocity.ini`). A server whose session storage differs from the one the cache recorded stores nothing for signed-in visitors, and says so in `X-Exp-Cache`.
3. **Restart Velocity** (`exp:velocity restart`) so it loads the kernel classes and asks the cache.

The first request on a cached siteaccess writes the configuration the early exit reads (`var/<site>/cache/exphttpcache/contract.php`, with a generated key).

Settings

File `settings/httpcache.ini`, block `[HttpCacheSettings]`, scope: installation. Override in `settings/override/httpcache.ini.append.php`.

Setting	Default	
`Enabled`	`disabled`	Nothing is stored or served while disabled. Switching off takes effect on the next request that reaches the kernel.
`CachedSiteAccesses[]`	`site`	Siteaccesses matched by host (`map`), by URI (`element`, `map`) or by `host_uri`; see "Which siteaccess" below.

Setting	Default	
`MaxAge`	`3600`	Longest life of a page even if nothing purges it. A template's `cache_ttl` shortens it; `cache_ttl=0` keeps the page out, as it does the view cache.
`StaleWhileRevalidate`	`60`	An expired (not purged) page is served as STALE while one request renders it again.
`ContentChangePurges`	`all`	`all`: any content change purges every page. `tags`: only pages showing the changed objects, nodes, parents, and pages with query blocks.
`QueryStringParameters[]`	none	Pages requested with any other query parameter are not cached.
`TagHeader`	`disabled`	The purge tags of each page in one header, for a purging proxy in front or while debugging: xkey (Varnish xkey vmod), `Surrogate-Key` (Fastly), `Cache-Tag` (Cloudflare). Off by default: the tags name internal ids. `ProxyHeaders=enabled`, the setting before it, still means xkey.
`APCu`	`enabled`	Hot entries in APCu in front of the files.
`MaxBodySize`	2 MB	Larger pages are not stored.

Which siteaccess, scheme and host

The early exit and Velocity answer before the kernel runs, so they work out the siteaccess, scheme and host themselves, from values the kernel writes into `contract.php`:

- **Siteaccess** (`ezpHttpCacheContract::resolveSiteAccess()`): the rules of `eZSiteAccess::match()` on the `site.ini` settings without siteaccess overrides -- `StaticMatch`, then `MatchOrder` with `uri` (`URIMatchType` `element` or `map`), `host` (`HostMatchType=map`) and `host_uri` (`HostUriMatchMapItems`, every host match method), then `DefaultAccess`. A request that reaches a rule it cannot follow (`port`, `servervar`, `index`, the `text` and `regexp` types, a name the kernel would normalise) is not served early. `/bold_ger/kontakt` on a `MatchOrder=uri;host` site is `bold_ger`, as the kernel has it.
- **Scheme and host** (`ezpHttpCacheContract::requestOrigin()`): as `eZSys::hostname()` and `eZSys::isSSLNow()` -- `X-Forwarded-Host`, then `Host`; `HTTPS`, the port against `SSLPort`, `X-Forwarded-Proto`, `X-Forwarded-Port`, `X-Forwarded-Server`. Behind a load balancer that ends TLS and forwards to `exp:8080`, a page is kept for `https://www.example.org/...`, the address the visitor asked for, and found under it again. The early exit reads `$_SERVER`; Velocity's server process needs an engine that hands over the request headers (Exponential Velocity v0.0.4.35 or later) and otherwise misses behind a load balancer.
- `RemoveSiteAccessIfDefaultAccess` **and** `PathPrefix` need nothing of their own. A page is kept under the full request URI, as the visitor sent it, so `/kontakt` (the default siteaccess, its name left out of links) and `/bold_ger/kontakt` are two entries, each found again by the URI that made it; `PathPrefix` changes which node a URI shows, not which siteaccess or URI the page is kept under.

The kernel stores a page only when these give the siteaccess, scheme and host it rendered the page with (`X-Exp-Cache: BYPASS (siteaccess not known before the kernel)` or `BYPASS (scheme or host not known before the kernel)` otherwise). A rule followed differently here costs a render, never a page of another siteaccess, scheme or host.

When pages are purged

- **An edit, publish, move, hide or delete** — the view cache clear the kernel already does fires `content/cache`; with `ContentChangePurges=all` every page goes, with `tags` the pages tagged with what changed: the tags Ibexa's HTTP cache purges, `l`, `pl` and `rl` of every node, `c` and `r` of every object. Exponential's own pages carry no `r` or `rl` tag; those two purge pages an extension tagged with them (`ezpHttpCacheListener::addTags()`) the Ibexa way. The purge is this cache's own: nothing is sent to a proxy in front (`TagHeader` only shows the tags). More than 10000 tags purged within `MaxAge` purge every page instead, so the state every lookup reads stays small. Scripts and cronjobs purge too (`eZScript` attaches the listeners).
- **Layout editor** — any successful write request to an `explayouts*` module purges every page. Layout blocks tag the page with every item they show and, for query blocks, `dq`, which every content change purges.
- **Roles, policies, assignments** — every context changes (a new generation).
- **A user's own data** — that user's context record is forgotten.
- **Setup > Caches,** `bin/php/ezcache.php` — "HTTP cache (role-aware pages)", `--clear-id=exphttpcache`, and with `--clear-tag=content` or `template`.

What else a page shows is tagged while it renders, pagelayout included: `fetch( 'content', 'list' )` and `list_count` of a parent's children tag `pl<parent>` (any change to one of them purges it), deeper `tree` fetches tag `dq`, and `fetch( 'content', 'node' )` tags `l<node>`. Listings inside a content view that comes from the view cache are not fetched again, and follow the view cache: the page is purged when that view cache is. Operators that read content some other way are not tagged, which is why `ContentChangePurges=all` stays the default; `tags` is for installations that know their templates.

Seeing what it does

- Every response says `X-Exp-Cache: HIT`, `STALE`, `MISS (reason)` or `BYPASS (reason)`. Under Velocity, a page answered by the server's own response cache (anonymous visitors) also says `X-Cache: HIT`; its `X-Exp-Cache` is the one stored with it, often `MISS` from the render that filled both caches.
- **Setup > System information** has an "HTTP cache (role-aware)" box: hit rate across all servers, why requests missed, entries, disk use, user contexts, the last purge, the settings, and buttons to purge everything, remove dead entries and reset the counters.
- The `cache_cleanup` cronjob part removes expired, purged and orphaned files (`cronjobs/httpcache_cleanup.php`).

What is never cached

POST and other writes; responses other than 200; responses that set a cookie of their own; pages with a disallowed query string; pages whose template sets `cache_ttl=0`; siteaccesses not in `CachedSiteAccesses`, or found by a rule that cannot be known before the kernel; anything not a content view; a request request-shield in front marked as not for a cache (`REQUEST_SHIELD=allow-uncached`: a made-up path or parameter); a page that carries the debug report of its request (the Exp Debug bar or the classic report, `X-Exp-Cache: BYPASS (debug report)`): its timings, queries, messages and the viewer's own panel height belong to

that request, and a report shown by IP would otherwise reach everyone sharing the anonymous context. A visitor the bar is shown to by address gets the page from the cache all the same, with the bar of its render (next section).

Visitors the debug bar is shown to

A visitor the Exp Debug bar is shown to by address ([DebugSettings] `DebugByIP=enabled`, `DebugIPList[]`, with `DebugOutput=enabled`) is answered from this cache as well, with the bar of the render that stored the page:

- The kernel stores the page such a visitor sees **without** the bar, under a context of that visitor's own (their role context plus their user ID), and the bar beside it in a file only the owner may read (0600: it holds SQL, settings and messages). The bar goes with the entry: the same tags purge it, and garbage collection removes it.
- On the next request the early exit (Apache) or the worker of Velocity puts the page and that bar together. The bar's first chip says **Cached**, its title when the page was stored and how long its render took, and a **Render now** link asks for a full render. The page time chip gives the time of this answer; every tab begins with "From the render at 12:03:15 (41 ms)", and the Timing tab with the time of this request. Every other figure of every tab is the render's, unchanged. The data block (`#exp-debug-data`) says the same: `served.by` is `cache`, `served.render_time` and `served.stored_at` describe the render, `summary.time` is this answer's.
- A page the kernel rendered says **Rendered** and why in the chip's title: no page in the cache yet, the page was purged, or a full render was asked for. Such a page is stored again, with its bar.
- A stale page is never served to such a visitor: within the stale-while-revalidate window they get a fresh render.

The full-render switch is the view parameter `(exp_render)/full`, added to any address: `/fitness/(exp_render)/full`, `/(exp_render)/full` for the front page, `/content/view/full/89/(exp_render)/full`. The kernel takes it out of the address before anything reads it, so it never reaches a URL alias, the view parameters, the content view cache, a cache-block key, a template or a link of the page, and the HTTP cache key never holds it. It means something only for a visitor the bar is shown to; for anybody else the address is the page without it. A page rendered with the switch has a **Back to the cached view** link.

The server process of Velocity, which asks this cache before a worker, does not know the visitor's address: for a user who has pages with the bar it leaves the request to a worker, which asks again with the address ("debug viewer, address not known here"). Visitors shown the bar by user ID only (`DebugByUser`) are rendered as before.

With request-shield in front

request-shield runs before `config.php` (`auto_prepend_file`, or the first line of `index.php`). Its HTTP cache stays off on Exponential 6 — this one is the page cache — and the two work together:

- **Never kept:** a request request-shield marks `allow-uncached` (a path or query parameter outside its cacheable definition) is answered and never stored: random addresses cannot fill the cache (`X-Exp-Cache: BYPASS (request-shield: not for a cache)`).

- **Fewer keys:** with `cache-ignore utm_* gclid …` request-shield takes tracking parameters out of `REQUEST_URI` before the early exit runs, so a campaign link is the same entry as the page.
- **Answered from the page without a made-up parameter:** with `set cache-unknown-query hit-only` request-shield names, in `REQUEST_SHIELD_CACHE_LOOKUP`, the address a request with a parameter no key holds may be answered from. The early exit takes it only when it names the request's own path and exactly the request's own parameters that `QueryStringParameters[]` lists, in any order (`X-Exp-Cache: HIT (request-shield lookup)`); a shield whose `cache-query` leaves one of them out gets a miss, never another page of a list. Nothing is stored for the request's own address, the answer says `Cache-Control: private` so a proxy in front keeps none either, and an expired page is served `STALE` without the lookup taking its refresh.
- **Its statistics** read `X-Exp-Cache` (HIT, STALE, MISS, BYPASS, each possibly followed by a reason in brackets): the response times by hit and miss in its dashboard.

Under Velocity the server process answers from this cache before a worker, and so before request-shield, which runs inside the worker from the first line of `index.php`: it sees the address as the visitor sent it, with no `cache-ignore` and no `REQUEST_SHIELD_CACHE_LOOKUP`, so neither of those two saves a render there. The early exit does not run in a worker at all. What does apply is **Never kept**: the listener that stores the page runs in the worker after request-shield, so an `allow-uncached` page is not stored under Velocity either. Velocity keeps `$_SERVER` entries whose name starts with `REQUEST_` from one request to the next in a persistent worker; switch request-shield off with `set mode off` (it still sets `REQUEST_SHIELD` on every request), not by removing the include, or a worker keeps the last request's `allow-uncached` and stores nothing until it is restarted.

Headers an extension adds to a page

A page answered from this cache is assembled without the kernel, from the headers stored with it: the content type and the security headers. A header an extension adds for one page (a content license as `Link` and `X-Content-License`, for example) would be missing on every hit. From 2026-10-11 the listener passes the headers it is about to store through the filter event `httpcache/store/headers`; a listener returns them with its own added:

INI

```ini
# extension/myext/settings/site.ini.append.php
[Event]
Listeners[]=httpcache/store/headers@myExtHttpCacheHeaders::filter
```

PHP

```php
class myExtHttpCacheHeaders
{
    public static function filter( array $headers )
    {
        $headers['X-My-Header'] = 'value for this page';
        return $headers;
    }
}
```

A listener that returns something other than an array is ignored. The headers are stored with the entry, so a hit carries them like the rendered page; keep them free of anything personal,

because every visitor of the same permission context gets the same copy. Filter events are listed in RAD extension points.

Security

- Keys and contexts are HMACs with a generated key (`secret`, 0600), never sent.
- A visitor with a session cookie the cache cannot resolve is a miss, never the anonymous page; an unknown session cookie name is a miss.
- A signed-in visitor gets pages of their own permission context only; a role change moves every user to a new context at once.
- Form tokens are per visitor and put back on every hit; no other personal data may be in a cached page — pages that show the visitor's name belong in a private context or out of the cache (`cache_ttl=0`).
- Files written by root (a CLI clear) are handed to the owner of the cache directory, so the web server can still update them.

Files

`kernel/private/classes/httpcache/ezphttpcachecontract.php`	Keys, contexts, entries, state, sessions, placeholders, serving; pure PHP
`kernel/private/classes/httpcache/ezphttpcacheearlyexit.php`	The early exit included from `config.php`
`kernel/private/classes/httpcache/ezphttpcachelistener.php`	Store after render, purges, sign-in record
`settings/httpcache.ini`	Settings
`cronjobs/httpcache_cleanup.php`	Removes dead entries
`tests/tests/kernel/classes/httpcache/ezpHttpCacheContractTest.php`	Unit tests (HC-01 … HC-16)

Related pages

- Behaviour changes, 16 to 30 September 2026: HTTP cache (compression once, headers on cached pages, siteaccess matching)
- Security defaults
- Velocity
- HTTP caching for anonymous visitors and Cache control from the console
- Chronicle, 16 to 30 September 2026
- Changelog 6.0.15
- Cache clears that move directories aside
- Velocity response cache

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.