

Purge

This explains the content cache purge mechanism, while publishing a content from the admin interface or from a container aware script (will not work with pure `eZScript` atm, but will with new CLI scripts).

Note that the approach is fully compatible with native reverse proxies like **Varnish** (but in this case you won't use all `HttpCache` related code).

Description

On publish, one or several `Http PURGE` requests are sent to the backend. This request will have a specific header `X-Location-Id` (in the case of 1 request per location to purge) or `X-Group-Location-Id` (in the case of 1 request for all location to purge).

Http PURGE requests

Emulated purge requests

By default, `PURGE` requests will be emulated and sent to the cache Store. Cache purge will thus be synchronous. This is the default behavior. **No `Http` requests will be sent to the backend when publishing.**

Configuration:

ezpublish.yml

```
ezpublish:
  http_cache:
    purge_type: local
```

One purge request for all locations, aka "SingleHttp"

Only one `Http PURGE` request is sent for purging every needed locations.

A request for purging locations 123 and 456 would be:

```
PURGE / HTTP 1.1
Host: localhost
X-Group-Location-Id: 123; 456
```

Configuration

ezpublish.yml

```
ezpublish:
  http_cache:
    purge_type: single_http
```

One request per location, aka "MultipleHttp"

One `Http PURGE` request will be sent *per location to purge*

A request for purging locations 123 and 456 would result to:

```
PURGE / HTTP 1.1
Host: localhost
X-Location-Id: 123
```

AND

```
PURGE / HTTP 1.1
Host: localhost
X-Location-Id: 456
```

Configuration:

ezpublish.yml

```
ezpublish:
  http_cache:
    purge_type: multiple_http
```

Purge all content

When purging all cached content, a single Http PURGE request is sent (except for *local* purge type), with *X-Location-Id* set to ***:

```
PURGE / HTTP 1.1
Host: localhost
X-Location-Id: *
```

Multiple servers

If you need to purge several servers at once (e.g. multiple Varnish infrastructure), you can set this up in the *siteaccess* configuration:

ezpublish.yml

```
ezpublish:
  http_cache:
    purge_type: single_http

  system:
    my_siteaccess:
      http_cache:
        purge_servers: ["http://varnish.server1/", "http://varnish.server2/",
"http://varnish.server3/"]
```

Manual purging

Manual purging is also possible:

```
$locationIds = array( 123, 456 );
$container->get( 'ezpublish.http_cache.purger' )->purge( $locationIds );
```