# Purge

This page explains the content cache purge *(aka invalidate)* mechanism used when publishing content from the UI or from a container-aware script, resulting in cache being invalidated either in the built-in Symfony Reverse Proxy, or on the much faster Varnish reverse proxy.

## Overview

As explained on the HttpCache page, eZ Platform returns content-related responses with an `X-Location-Id` header that are stored together by the configured HTTP cache. This allows you to clear *(invalidate)* HTTP cache representing specifically a given Content item. On publishing the content, a cache purger is triggered with the Content ID in question, which in turn figures out affected content Locations based on Smart HTTP cache clearing logic. The returned Location IDs are sent for purge using the purge type explained further below.

## Purge types

### Symfony Proxy: Local purge type

By default, invalidation requests will be emulated and sent to the Symfony Proxy cache Store. Cache purge will thus be synchronous, meaning no HTTP purge requests will be sent around when publishing.

**ezplatform.yml**

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

### Varnish: HTTP purge type

With Varnish you can configure one or several servers that should be purged over HTTP. This purge type is asynchronous, and flushed by the end of Symfony kernel-request/console cycle *(during terminate event)*. Settings for purge servers can be configured per site group or site access:

**ezplatform.yml**

```
ezpublish:
    http_cache:
        purge_type: http

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

For further information on setting up Varnish, see Using Varnish.

## Purging

While purging on content, updates are handled for you; on actions against the eZ Platform APIs, there are times you might have to purge manually.

### Manual by code

Manual purging from code which takes Smart HttpCache clearing logic into account, this is using the service also used internally for cache clearing on content updates:

```
// Purging cache based on content id, this will trigger cache clear of all locations
found by Smart HttpCache clear
// typically self, parent, related, ..
$container->get('ezpublish.http_cache.purger')->purgeForContent(55);
```

### Manually by command with Symfony Proxy

Symfony Proxy stores its cache in the Symfony cache directory, so a regular `cache:clear` commands will clear it:

```
php app/console --env=prod cache:clear
```

### Manual by HTTP BAN request on Varnish

When using Varnish and in need to purge content directly, then the following examples show how this is done internally by our FOSPurgeClient, and in turn FOSHttpCache Varnish proxy client:

For purging all:

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

Or with given location ids *(here 123 and 234)*:

```
BAN / HTTP 1.1
Host: localhost
X-Location-Id: ^(123|234)$
```