Persistence cache configuration

Introduction

Tech Note

Current implementation uses a caching library called Stash (, via Stash-bundle). Stash supports the following cache backends: **FileSystem**, **Memcache**, **APC**, **Sqlite, Redis** and **BlackHole.**

Use of Memcached or Redis is a requirement for use in Clustering setup, for overview of clustering feature see Clustering.

If eZ Publish Platform changes to another cache system, configuration will change in the future, changes to configuration in StashBundle is listed here:

> **Configuration change in 5.4/2014.07**
> StashBundle version bundled with 5.4/2014.07 and higher refers to cache backends as *"drivers"* where it was previously referred to as *"handlers"* in yml configuration

Cache service

The cache system is exposed as a "cache" service, and can be reused by any other service as described on the Persistence cache page.

Configuration

During Setup wizard and manually using `ezpublish:configure` console command a default configuration is generated currently **using FileSystem**, using `%kernel.cache_dir%/stash` to store cache files.

The configuration is placed in ezpublish/config/ezpublish.yml, and looks like:

Default ezpublish.yml

```
stash:
    caches:
        default:
            # For eZ Publish Platform versions prior to 5.4/2014.07, use "handlers"
instead of "drivers"!
            drivers:
                - FileSystem
            inMemory: false
            registerDoctrineAdapter: false
```

The default settings used during setup wizard as found in `ezpublish/config/ezpublish_setup.yml`:

ezpublish_setup.yml

```
stash:
    caches:
        default:
            # For eZ Publish Platform versions prior to 5.4/2014.07, use "handlers"
instead of "drivers"!
            drivers:
                - BlackHole
            inMemory: true
            registerDoctrineAdapter: false
```

This setting works across all installs and just caches objects within the same request thanks to the `inMemory: true` setting.

If you want to change to another cache backend, see in *Stash backend configuration* below for what kind of settings you have available.

> **Note for "inMemory" cache with long running scripts**
> Use of `inMemory` caching with BlackHole or any other cache backend should not be used for long running scripts as it will over time return stale data, inMemory cache is not shared across requests/processes, so invalidation does not happen!

Multi repository setup

New in 5.2 is the possibility to select a specific Stash cache pool on a siteaccess or sitegroup level, the following example shows use in a sitegroup:

ezpublish.yml site group setting

```
ezdemo_group:
            cache_pool_name: "default"
            database:
                ...
```

The "default" here refers to the name of the cache pool as specified in the *stash* configuration block shown above, if your install has several repositories (databases), then make sure every group of sites using different repositories also uses a different cache pool to avoid unwanted effects.
NB: We plan to make this more native in the future, so this setting will someday not be needed.

Stash cache backend configuration

General settings

To check which cache settings are available for your installation, run the following command in your terminal :

```
php ezpublish/console config:dump-reference stash
```

FileSystem

This cache backend is using local filesystem, by default the Symfony cache folder, as this is per server, it **does not support multi server (cluster) setups**!

We strongly discourage you from storing cache files on NFS, as it defeats the purpose of the cache: speed

Available settings

`path`	The path where the cache is placed, default is `%kernel.cache_dir%/stash`, effectively `ezpublish/cache/<env>/stash`
`dirSplit`	Number of times the cache key should be split up to avoid having to many files in each folder, default is 2.
`filePermissions`	The permissions of the cache file, default is 0660.
`dirPermissions`	The permission of the cache file directories (see dirSplit), default is 0770.
`memKeyLimit`	Limit on how many key to path entries are kept in memory during execution at a time to avoid having to recalculate the path on key lookups, default 200.
`keyHashFunction`	Algorithm used for creating paths, default md5. Use crc32 on Windows to avoid path length issues.

Issues with Microsoft Windows
If you are using a Windows OS, you may encounter an issue regarding **long paths for cache directory name**. The paths are long because Stash uses md5 to generate unique key that are sanitized really quickly.

Solution is to **change the hash algorithm** used by Stash.

Specifying key hash function

```
stash:
    caches:
        default:
            # For eZ Publish Platform versions prior to 5.4/2014.07, use
"handlers" instead of "drivers"!
            drivers:
                - FileSystem
            inMemory: true
            registerDoctrineAdapter: false
            FileSystem:
                keyHashFunction: 'crc32'
```

This configuration is only recommended for Windows users.

Note: You can also define the **path** where you want the cache files to be generated to be able to get even shorter system path for cache files.

FileSystem cache backend troubleshooting

By default, Stash Filesystem cache backend stores cache to a sub-folder named after the environment (`i.e. ezpublish/cache/dev`, `ezpublish/cache/prod`). This can lead to the following issue : if different environments are used for operations, persistence cache (manipulating content, mostly) will be affected and cache can become inconsistent.

To prevent this, there are 2 solutions :

1. ***Manual***

   Always use the same environment, for web, command line, cronjobs...

2. ***Share stash cache across Symfony environments (prod / dev)***

Either by using another Stash cache backend, or by setting Stash to use a shared cache folder that does not depend on the environment.
In ezpublish.yml:

```
stash:
    caches:
        default:
            FileSystem:
                path: "%kernel.root_dir%/cache/common"
```

This will store stash cache to `ezpublish/cache/common.`

APC

This cache backend is using shard memory using APC's user cache feature, as this is per server, it **does not support multi server (cluster) setups**.

> **Limitation**
> As APC user cache is not shared between processes, it is not possible to clear the user cache from CLI, even if you set `apc.enable_cli` to On. Hence publishing content from a command line script won't let you properly clear SPI Persistence cache.

Please also note that the default value for `apc.shm_size` is 128MB. However, 256MB is recommended for APC to work properly. For more details please refer to the APC configuration manual.

Available settings

`ttl`	The time to live of the cache in seconds, default set to 500 (8.3 minutes)
`namespace`	A namespace to prefix cache keys with to avoid key conflicts with other eZ Publish sites on same eZ Publish installation, default is `null`.

Memcache

This cache backend is using Memcached, a distributed caching solution, this is the only supported cache solution for multi server (cluster) setups!

> **Note**
> Stash supports both the php-memcache and php-memcached extensions. **However** only php-memcache is officially tested on Redhat/Centos while php-memcached is on Debian and Ubuntu. If you have both extensions installed, Stash will automatically choose php-memcached.

`servers`	Array of Memcached servers, with host/IP, port and weight **`server`**: Host or IP of your Memcached server **`port`**: Port where Memcached is listening to (defaults to 11211) **`weight`**: Weight of the server, when using several Memcached servers
`prefix_key`	A namespace to prefix cache keys with to avoid key conflicts with other eZ Publish sites on same eZ Publish installation (default is an empty string). Must be the same on all server with the same installation. See Memcached prefix_key option.
`compression`	default true. See Memcached compression option.
`libketama_compatible`	default false. See Memcached libketama_compatible option
`buffer_writes`	default false. See Memcached buffer_writes option
`binary_protocol`	default false. See Memcached binary_protocol option

no_block	default false. See Memcached no_block option
tcp_nodelay	default false. See Memcached tcp_nodelay option
connection_timeout	default 1000. See Memcached connection_timeout option
retry_timeout	default 0. See Memcached retry_timeout option
send_timeout	default 0. See Memcached send_timeout option
recv_timeout	default 0. See Memcached recv_timeout option
poll_timeout	default 1000. See Memcached poll_timeout option
cache_lookups	default false. See Memcached cache_lookups option
server_failure_limit	default 0. See PHP Memcached documentation
socket_send_size	See Memcached socket_send_size option.
socket_recv_size	See Memcached socket_recv_size option.
serializer	See Memcached serializer option.
hash	See Memcached hash option.
distribution	Specifies the method of distributing item keys to the servers. See Memcached distribution option.

All settings but *servers* are only available with memcached php extension, for more information on these settings and which version of php-memcached they are available in, see: http://php.net/Memcached
If you are on eZ Publish 5.1, make sure to update Stash and StashBundle to get access to these settings.

Example with Memcache

```
stash:
    caches:
        default:
            # For eZ Publish Platform versions prior to 5.4/2014.07, use "handlers"
instead of "drivers"!
            drivers: [ Memcache ]
            inMemory: true
            registerDoctrineAdapter: false
            Memcache:
                prefix_key: ezdemo_
                retry_timeout: 1
                servers:
                    -
                        server: 127.0.0.1
                        port: 11211
```

Connection errors issue
If memcached does display connection errors when using the default (ascii) protocol, switching to binary protocol (in the stash configuration and memcached daemon) should resolve the issue.

Memory consumption issues with Redis

>= 5.4.13

If you use the Redis cache driver and encounter problems with high memory consumption, you can use the following *(non-SiteAccess-aware)* settings:

ezpublish.yml

```
ezpublish:
    stash_cache:
        igbinary: true/false
        lzf: true/false
```

- `ezpublish.stash_cache.igbinary` enables you to use the igbinary PHP extension to serialize objects stored in cache.
  - *Recommended enabled in most use cases.*
- `ezpublish.stash_cache.lzf` enables you to use the LZF PHP extension to compress serialized objects stored in cache.
  - *Compression consumes some more CPU. A scenario where it improves overall performance is when Redis memory and/or network bandwidth is limited, with spare CPU capacity on PHP server.*

These options in combination results in around 1/2 memory usage for API caching compared to not being enabled, igbinary accounts for ~75% of that and LZF the remaining ~25% when configured for max compression.

After changing these settings you need to clear cache and purge Redis content .