The request lifecycle

What happens between a browser request and the response in Exponential, step by step - maintenance and repair hooks, autoload, the kernel constructor, siteaccess matching, URL translation, access checks, the module run...

Source file: `doc/eng-us/help/software-engineers/request-lifecycle.md` Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/help/software-engineers/request-lifecycle.md

This page follows one web request through Exponential, from `index.php` to the response, and names the class and file of each step. It is the summary of The request lifecycle in the kernel architecture standard; read that section when you change anything on this path.

Contents

1. Before the kernel
2. The kernel constructor
3. The dispatch loop
4. Output and shutdown
5. Short cuts: who can answer first
6. The same code outside a web request
7. Rules for changes on this path

Before the kernel

1. **Maintenance and repair hooks.** If `var/maintenance.json` exists, `expMaintenance::check()` (`kernel/classes/expmaintenance.php`) may answer the request with the maintenance page. A request carrying `exp_repair` is handed to `ezpRepairQueue` (`lib/ezutils/classes/ezprepairqueue.php`), which can repair an installation without its Composer libraries (Repair from the browser).
2. **Autoload.** `autoload.php` loads the generated class maps: `autoload/ezp_kernel.php` and the extension map written by `php bin/php/ezpgenerateautoloads.php -e`.
3. **Kernel object.** `new ezpKernel( new ezpKernelWeb() )`, then `$kernel->run()->getContent()`. `ezpKernel` is a thin wrapper around a handler implementing `ezpWebBasedKernelHandler`.

The kernel constructor

`ezpKernelWeb` (`kernel/private/classes/ezpkernelweb.php`), in order:

Step	What happens
Settings	reads `i18n.ini` and `site.ini`, sets the time zone and file permissions, initializes eZSys
Global extensions	`eZExtension::activateExtensions( 'default' )`
Siteaccess	`eZSiteAccess::match()` on the request URI, unless one was injected, then `eZSiteAccess::change()`

Step	What happens
Siteaccess extensions	`eZExtension::activateExtensions( 'access' )`, and the INI instances are reset so siteaccess settings apply

The dispatch loop

`run()` and `dispatchLoop()`:

1. Session and request initialization (`sessionInit()`, `requestInit()`).
2. The events `request/preinput` and `request/input` (Events, extension points and plugins).
3. URL translation: a URL alias or wildcard becomes a module address (`eZURLAliasML`, `eZURLWildcard`); `/About-us` becomes, for example, `content/view/full/58`.
4. The module check (`eZModule::accessAllowed()`), `fetchModule()`, the siteaccess login check (the `user/login` policy) and the view access check (`eZUser::hasAccessToView()`).
5. Request rules, when configured, decide after URL translation and the policy check whether to allow, deny, redirect, rewrite or log (Request rules).
6. `$module->run( $functionName, $params, ... )` runs the view. A view can ask for a rerun with another URI or a redirect (`eZModule::STATUS_RERUN`, `STATUS_REDIRECT`); the loop handles both.

Output and shutdown

- The view's `$Result` is placed into the pagelayout (`pagelayout.tpl` unless the view sets another), passed through `eZDisplayResult()` and the output filters `response/preoutput` and `response/output` (`kernel/private/classes/global_functions.php`).
- `shutdown()` writes pending data. Under Velocity the worker is then reset for the next request (Velocity persistent workers).
- Every request is classified once, whichever way it ends, and a not-found answer or a probe goes to its own log (Request logs).

Short cuts: who can answer first

Who	When	Kernel started?
Browser or CDN	the page or file is still fresh in their cache	no request at all
Velocity response cache	a public page is cached in the server process	no
Role-aware HTTP cache	enabled, and a stored view matches the permission context	no (an early exit from `config.php` under PHP-FPM)
Maintenance hook	maintenance mode is on	partly (before the kernel)
Content view cache	a cached copy of this content view exists	yes, but the view is not rendered

Details: Caching layers for engineers.

The same code outside a web request

Command line scripts and cronjobs do not use `ezpKernelWeb`: they boot through `eZScript`. The console (`bin/php/console`) discovers commands; `runcronjobs.php` runs cronjob parts (Console, cronjobs and background work).

Rules for changes on this path

- A change **must** keep working under PHP-FPM, Velocity and FrankenPHP.
- Code in `ezpKernelWeb` guards newer classes with `class_exists()`, because a long-running Velocity worker may have loaded its class map before the class existed; new kernel hooks should do the same.
- Per-request state **must not** live in static properties without a reset.
- No `exit` for flow control except through `eZExecution::cleanExit()`.

Related pages

- Architecture overview
- Modules, views and access control
- Concurrency, statics and long-lived processes
- 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.