REST and remote services from PHP

The HTTP APIs of Exponential from the PHP side - the REST kernel behind index_rest.php, rest.ini, providers implementing ezpRestProviderInterface with routes and ezpRestMvcController controllers, authentication with...

Source file: `doc/eng-us/help/php-developers/rest-and-remote-services-from-php.md` Last changed 2026-11-07
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/rest-and-remote-services-from-php.md

This page is for PHP developers who serve data to other programs: mobile apps, other front ends, integrations. Exponential has three HTTP layers for that. This page explains how they are built in PHP and how to add to the REST one; the consumer view is REST API v1 and v2 for applications.

Contents

1. Three layers
2. The REST kernel
3. Adding a provider
4. Controllers
5. Authentication
6. Authorization is your job
7. Choosing a layer

Three layers

Layer	Entry	Built for
REST	`index_rest.php`, under `/api/` (`rest.ini [System] ApiPrefix=/api`)	content and resources for other programs, OAuth2 for third parties
expservices	`ezjscore/call/exp<domain>::<function>`	administration features from scripts and apps, with discovery and API tokens (Backend services over ezjscore)
ezjscore server functions	`ezjscore/call/<name>::<function>`	your own pages' JavaScript (ezjscore JSON endpoints in PHP)

The REST kernel

`index_rest.php` runs the REST kernel in `kernel/private/rest/` (`ezpkernelrest.php`, the router, the MVC configuration, renderers, OAuth). It is built on the MVC tools of the expcomponents (`se7enxweb/expcomponents-mvc-tools`, the fork of `zetacomponents/mvc-tools` since 6.0.15; expcomponents), so a provider supplies routes and controllers in that framework's shape. Settings are in `settings/rest.ini`:

Block	Holds
`[System]`	`ApiPrefix`, the prefix filter class
`[ApiProvider]`	`ProviderClass[<prefix>]=<class>`, one entry per provider (auth is `ezpRestAuthProvider`)
`[RouteSettings]`	the route filter (`RouteSettingImpl`) and routes that skip authentication
`[Authentication]`	whether requests must authenticate (`RequireAuthentication=enabled` by default)
`[OAuthSettings]`, `[ApiKeySettings]`	token lifetime and API keys

The content API (`/api/ezp/v1/...`, `/api/ezp/v2/...`) comes from the `ezprestapi` extension, whose provider is registered the same way.

Adding a provider

A provider implements `ezpRestProviderInterface` (`kernel/private/rest/classes/interfaces/rest_provider.php`):

Method	Returns
`getRoutes()`	the provider's routes, mapping URL patterns and HTTP methods to controller actions
`getViewController()`	the view controller that picks a renderer for the result

Register it for a prefix in your extension's `rest.ini.append.php`:

INI

```
<?php /* #?ini charset="utf-8"?
[ApiProvider]
ProviderClass[acme]=acmeRestProvider
*/ ?>
```

Setup > RAD has a tool that writes the skeleton (`/setup/handlerextension/restprovider`); route filters are a separate extension point (`[RouteSettings] RouteSettingImpl`) (Extension points: REST provider). Read the shipped providers in `kernel/private/rest/` and in `ezprestapi` before writing your own: their route tables are the best reference for the exact format.

Controllers

Controllers extend `ezpRestMvcController` (`kernel/private/rest/classes/controllers/rest_controller.php`), which extends the `ezcMvcController` of the MVC tools (the class name is unchanged in the fork). Each action reads its route variables and request, does the work through your service classes, and returns a result that the view controller renders as JSON. Keep actions thin, as for module views: the same service should be callable from a command and a test.

Authentication

Credential	For
OAuth2 access token (`POST /api/auth/v1/oauth/token`, grants `authorization_code` and `refresh_token`)	an application acting for its users; clients are registered in **Setup > oAuth admin**
personal API key (`Authorization: Bearer expk_...`)	a user's own script (Personal API keys)

Behind Apache with FastCGI, make sure the `Authorization` header reaches PHP (OAuth and API keys).

Authorization is your job

The REST framework **authenticates**: it tells you who the user is. It does not decide what the user may do. Every controller action must check policies for what it reads or writes. The content API does this through the kernel's `expRestContentPermission` (`content/read`, `content/create`, `content/edit`, `content/remove`) and answers `401` without valid credentials, `403` when the user may not, `404` for an unknown node. Do the same in your actions, with `eZUser::currentUser()->hasAccessTo()` and `canRead()`/`canEdit()` on content.

Choosing a layer

Need	Layer
a third party integrates with your content, with OAuth consent	REST
a mobile app reads and creates content	REST (content API)
a script automates administration tasks	expservices with an API token
your page's JavaScript needs a small answer	ezjscore server function
another Exponential installation syncs content	the `syndication` extension, when installed

Related pages

- REST API v1 and v2
- Remote services with expservices
- Headless Exponential
- 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.