PSR-CAS Documentation

Release 1.0.0

Pol Dellaiera

Jan 30, 2020

Contents

PSR CAS, a standard PHP library for CAS authentication.

The Central Authentication Service (CAS) is an Open-Source single sign-on protocol for the web. Its purpose is to permit a user to access multiple applications while providing their credentials only once. It also allows web applications to authenticate users without gaining access to a user's security credentials, such as a password. The name CAS also refers to a software package that implements this protocol.

For improving the flexibility and in order to maximize it, it is able to authenticate users and leaves the session handling up to the developer.

In order to foster a greater adoption of this library, it has been built with interoperability in mind. It only uses PHP Standards Recommendations interfaces.

- PSR-3 for logging,
- PSR-4 for classes autoloading,
- PSR-6 for caching,
- PSR-7 for HTTP messages (requests, responses),
- PSR-12 for coding standards,
- PSR-17 for HTTP messages factories,
- PSR-18 for HTTP client.

Therefore, this library is framework agnostic and can be integrated in any PHP project, with any framework.

CHAPTER 1

Requirements

1.1 PHP

PHP greater than 7.1 is required for this library.

1.2 PHP Extensions

- json
- libxml
- simplexml

1.3 Packages

In order to get the PSR CAS library running, you will require some dependencies.

To give a maximum freedom to the users using PSR CAS, each required dependencies is a well defined standardized PHP class.

Dependency	PSR	Implementations	Example package
Logger	PSR-3	log-implementation	monolog/monolog
Cache	PSR-6	cache-implementation	symfony/cache
Server request	PSR-7	http-message-implementations	nyholm/psr7-server
HTTP factories	PSR-17	http-factory-implementations	nyholm/psr7
HTTP Client	PSR-18	http-client-implementations	symfony/http-client

You are free to use any package you want, as long as they are implementing the proper requirement.

PSR CAS only returns standardized HTTP responses, you will need to emit the response back to the client.

You may use custom code for that, but you can also use any of the following packages for this

- zendframework/zend-httphandlerrunner
- http-interop/response-sender

CHAPTER 2

Installation

The easiest way to install it is through Composer

```
composer require drupol/psrcas
```

CHAPTER 3

Configuration

```
base_url: https://casserver.herokuapp.com/cas
protocol:
  login:
    path: /login
    allowed_parameters:
      - service
      - renew
      - gateway
  serviceValidate:
    path: /p3/serviceValidate
    allowed_parameters:
      - service
      - ticket
      - pgtUrl
      - renew
      - format
    default_parameters:
      pgtUrl: https://my-app/casProxyCallback
  logout:
    path: /logout
    allowed_parameters:
      - service
    default_parameters:
      service: https://my-app/homepage
  proxy:
    path: /proxy
    allowed_parameters:
      - targetService
      - pgt
  proxyValidate:
    path: /proxyValidate
    allowed_parameters:
      - service
      - ticket
```

(continues on next page)

(continued from previous page)

```
    - pgtUrl
  default_parameters:
    pgtUrl: https://my-app/casProxyCallback
```

CHAPTER 4

Usage

Apereo already provides a demo CAS server without no proxy authentication mechanism enabled.

In order to test the libraries here, I've setup another CAS server with Proxy authentication enabled this time.

Feel free to use it for your tests.

> **Warning:** If your client application is not hosted on a public server and in HTTPS, this won't work.

Tip: See more on the page *Development*. if you want to have your own local CAS server.

The test login is *casuser*, password is: *Mellon*

4.1 Bare PHP

To get you started with PSR CAS in a simple bare PHP project (*without using any framework*), you can check the following project: drupol/psrcas-client-poc

Test the bare PHP demo application now.

4.2 Symfony

The PSR CAS library can be used in a Symfony (4 or 5) project through the package drupol/cas-bundle

Test the Symfony bundle demo application now.

See the documentation of the PSR CAS Bundle for more information.

CHAPTER 5

Tests, code quality and code style

Every time changes are introduced into the library, Travis CI and Github Actions run the tests written with PHPSpec.

PHPInfection is also triggered used to ensure that your code is properly tested.

The code style is based on PSR-12 plus a set of custom rules. Find more about the code style in use in the package drupol/php-conventions.

A PHP quality tool, Grumphp, is used to orchestrate all these tasks at each commit on the local machine, but also on the continuous integration tools (Travis, Github actions)

To run the whole tests tasks locally, do

```
composer grumphp
```

or

```
./vendor/bin/grumphp run
```

Here's an example of output that shows all the tasks that are setup in Grumphp and that will check your code

```
$ ./vendor/bin/grumphp run
GrumPHP is sniffing your code!
Running task  1/13: SecurityChecker... ✓
Running task  2/13: Composer... ✓
Running task  3/13: ComposerNormalize... ✓
Running task  4/13: YamlLint... ✓
Running task  5/13: JsonLint... ✓
Running task  6/13: PhpLint... ✓
Running task  7/13: TwigCs... ✓
Running task  8/13: PhpCsAutoFixerV2... ✓
Running task  9/13: PhpCsFixerV2... ✓
Running task 10/13: Phpcs... ✓
Running task 11/13: PhpStan... ✓
Running task 12/13: Phpspec... ✓
Running task 13/13: Infection... ✓
$
```

CHAPTER 6

Contributing

See the file CONTRIBUTING.md but feel free to contribute to this library by sending Github pull requests.

CHAPTER 7

Development

In order to test efficiently, is to test the library against a real CAS server.

If you're not able to use one, the best is to work with a local CAS server.

If you want to setup your own local CAS server in less than 2 minutes, use the repo crpeck/cas-overlay-docker and you'll have something working really quickly.

Don't forget to setup the HTTPS certificates because the communication between the CAS server and your application MUST be in HTTPS, and I haven't found a way yet to disable this for testing purposes.

If you prefer to use your local machine, there are already some documentation on Github.