EU-Login-bundle Documentation

Release 1.0.0

May 30, 2023

Contents

A Central Authentication Service bundle for Symfony.

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.

In order to foster a greater adoption of this bundle, 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.

CHAPTER 1

Requirements

1.1 PHP

PHP greater or equal to 7.1.3 for Symfony 4.

PHP greater or equal to 7.2.5 for Symfony 5.

1.2 Symfony

The minimal required version of Symfony is 4.

CHAPTER 2

Installation

This package has a Symfony Flex recipe that will install configuration files for you.

Default configuration files will be copied in the *dev* environment.

2.1 Step 1

The easiest way to install it is through Composer

```
composer require ecphp/eu-login-bundle
```

2.2 Step 2

Edit the configuration file *config/packages/dev/cas_bundle.yaml* and make the necessary changes to fit your needs.

See more on the dedicated *Configuration* page.

2.3 Step 3

Read the ecphp/cas-bundle documentation to have more information on how to enable a firewall and protect your application.

CHAPTER 3

Configuration

```yaml
cas:
    base_url: https://webgate.ec.europa.eu/cas
    protocol:
        login:
            path: /login
            default_parameters:
                service: cas_bundle_homepage
        serviceValidate:
            path: /serviceValidate
            default_parameters:
                userDetails: "true"
                #pgtUrl: cas_bundle_proxy_callback
        logout:
            path: /logout
            default_parameters:
                service: cas_bundle_homepage
        proxy:
            path: /proxy
        proxyValidate:
            path: /proxyValidate
            default_parameters:
                userDetails: "true"
                #pgtUrl: cas_bundle_proxy_callback
```

CHAPTER 4

Usage

Once the bundle installed and setup properly, browsing a secured path should redirect you to EU Login, straight for authentication.

CHAPTER 5

Tests, code quality and code style

Every time changes are introduced into the library, 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  1/10: Composer... ✓
Running task  2/10: ComposerNormalize... ✓
Running task  3/10: YamlLint... ✓
Running task  4/10: JsonLint... ✓
Running task  5/10: PhpLint... ✓
Running task  6/10: TwigCs... ✓
Running task  7/10: PhpCsAutoFixerV2... ✓
Running task  8/10: PhpCsFixerV2... ✓
Running task  9/10: Phpcs... ✓
Running task 10/10: PhpStan... ✓
$
```

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.

If you want to test against EU Login, make sure that your application respond on the `localhost` hostname, it's the only domain for which it will work. However, only basic authentication will work and it will not be possible to enable authentication with proxy because the hostname `localhost` will not be accessible from the Internet.

7.1 Maintainers

See the MAINTAINERS.txt file.

7.2 Contributors

See the Github insights page.