ECPHP - API Gateway Authentication Bundle documentation

Release 1.0.0

Apr 25, 2023

CONTENTS

This Symfony bundle provides the necessary to authenticate a request having a specific `Authorization` HTTP header with a `pop` token.

Supported tokens are:

- Opaque PoP access tokens issued by EU Login OpenID Connect (`pop [JWS signed AT...]`)

CHAPTER

ONE

REQUIREMENTS

1.1 PHP

PHP greater or equal to 7.4.

1.2 Symfony

The minimal required version of Symfony is 5.

1.3 Extensions

These PHP extensions are required:

- openssl
- gmp
- sodium

CHAPTER

TWO

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 recommended way to install it is with Composer :

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

This package has a Symfony recipe that will provides the minimum configuration files.

> **Warning:** Be careful, the recipe will enable some routes in your `dev` environment only. Those routes might be considered as a security issue if they are enabled in the `production` environment. Those routes are `/api/token` and `/api/user`. Find the documentation related to those routes inside the classes themselves. To disable them completely, just delete the file `packages/config/routes/dev/eu_login_api_authentication.yaml`.

2.2 Step 2

Edit the bundle configuration by editing the file `config/packages/dev/eu_login_api_authentication.yaml`.

```yaml
eu_login_api_authentication:
    client_id: foo
    client_secret: bar
    environment: acceptance # Available values are: acceptance, production
```

2.3 Step 3

This is the crucial part of your application's security configuration.

Edit the security settings of your application by edition the file *config/packages/security.yaml*.

```yaml
security:
    enable_authenticator_manager: true
    firewalls:
```

(continues on next page)

(continued from previous page)

```
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            custom_authenticators:
                - 'eu_login_api_authentication.authenticator'

    access_control:
        - { path: ^/user, role: IS_AUTHENTICATED_FULLY }
```

Feel free to change these configuration to fits your need. Have a look at the Symfony documentation about security and authentication.

CHAPTER

THREE

CONFIGURATION

Hereunder an example of configuration for this bundle.

```yaml
eu_login_api_authentication:
    client_id: foo
    client_secret: bar
    environment: acceptance
```

This bundle uses EU Login to authenticate the incoming requests.

In order to authenticate, you need to be able to create valid tokens through your frontend application.

Sometimes, the frontend application is not ready and you still need to be able to authenticate.

Basically, you need to generate and authenticate tokens without relying on EU Login.

In order to do that, follow the following steps:

1. Edit the content of your application `services.yaml` and add:

```yaml
when@dev:
    services:
        EcPhp\EuLoginApiAuthenticationBundle\Service\LocalEuLoginApiCredentials:
            decorates: 'eu_login_api_authentication.service'
            arguments: ['@.inner']
```

2. This will replace the EU Login authentication mechanism by another one which does not require any connection to EU Login.

   Warning: Be extremely careful to not enable that for production environment.

3. Read the official Symfony documentation if you want to enable this only for a particular environment.

CHAPTER

FOUR

USAGE

4.1 Step 1

Follow the *Installation* procedure.

4.2 Step 2

Configure the configuration files accordingly and the security of your Symfony application.

4.3 Step 3

Get a valid token from your front-end application.

4.4 Step 4

- Make a request to `/api/user` with the `Authorization` header.

```
curl -X GET "http://127.0.0.1:8000/api/user" -H "Authorization:  pop <insert-token-here>"
```

4.5 Step 5

Make sure that the development routes are enabled.

If they are not, create a new file `eu-login-api-authentication-bundle.yaml` in `config/packages/routes/dev/` with the following content:

```yaml
eu_login_api_authentication_bundle:
    resource: '@EuLoginApiAuthenticationBundle/Resources/config/routes/routes.php'
    prefix: /api
```

The routes `/api/token` and `/api/user` will be available.

Generate a basic token:

```
GET http://127.0.0.1:8000/api/token
```

And the response:

```
HTTP/1.1 200 OK
Cache-Control: no-cache, private
Content-Type: application/json
Date: Thu, 22 Apr 2021 13:31:44 GMT, Thu, 22 Apr 2021 13:31:44 GMT
Host: 127.0.0.1:8000
X-Debug-Token: 4ff71a
X-Debug-Token-Link: http://127.0.0.1:8000/_profiler/4ff71a
X-Powered-By: PHP/7.4.16
X-Robots-Tag: noindex
Content-Length: 288
Connection: close

{
"token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJhdCI6ImV5SjBlWEFpT2lKS1YxUWlMQ0poYkdjaU9pSklVekkxTmlKOS5leUp6ZFdJaU9pSjFjMlZ5WHpZd09ERTNZV013TUdWbl
↪nEPLVP34eSMge_qz9Jrw88_w6BQHzKKk6aeyj38F8rU"
}
```

Generate a basic token with custom fields:

```
POST http://127.0.0.1:8000/api/token
Content-Type: application/json

{ "key" : "value", "list": [1, 2, 3] }
```

and the response:

```
HTTP/1.1 200 OK
Cache-Control: no-cache, private
Content-Type: application/json
Date: Thu, 22 Apr 2021 13:32:38 GMT, Thu, 22 Apr 2021 13:32:38 GMT
Host: 127.0.0.1:8000
X-Debug-Token: 80b1ca
X-Debug-Token-Link: http://127.0.0.1:8000/_profiler/80b1ca
X-Powered-By: PHP/7.4.16
X-Robots-Tag: noindex
Content-Length: 340
Connection: close

{
"token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJhdCI6ImV5SjBlWEFpT2lKS1YxUWlMQ0poYkdjaU9pSklVekkxTmlKOS5leUpyWlhraU9pSjJZV3gxWlNJc0lteHBjM1FpT2xzel
↪HJY2L-oS09IqVI_q0SGGzarE6l6ZXHQAb14F-1STwzQ"
}
```

Use `/api/user` to introspect a token:

```
GET http://127.0.0.1:8000/api/user
Authorization: pop eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJhdCI6ImV5SjBlWEFpT2lKS1YxUWlMQ0poYkdjaU9pSklVekkxTmlKOS5leUp6ZFdJaU9pSjFjMlZ5WHpZd09ERTNPV1ppWVRVel
↪8MotNjUqlVgzKnAY4CGDm63TdmGrBsPf3_Jvjy_q3qs
```

And the response:

```
HTTP/1.1 200 OK
Cache-Control: no-cache, private
Content-Type: application/json
Date: Thu, 22 Apr 2021 13:29:33 GMT, Thu, 22 Apr 2021 13:29:33 GMT
Host: 127.0.0.1:8000
X-Debug-Token: 716819
X-Debug-Token-Link: http://127.0.0.1:8000/_profiler/716819
X-Powered-By: PHP/7.4.16
X-Robots-Tag: noindex
Content-Length: 42
Connection: close

{
"sub": "user_60817a347e064",
"active": true
}
```

Introspect a token having custom fields

```
GET http://127.0.0.1:8000/api/user
Authorization: pop eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
↪eyJhdCI6ImV5SjBlWEFpT2lKS1YxUWlMQ0poYkdjaU9pSklVekkxTmlKOS5leUpyWlhraU9pSjJZV3gxWlNJc0lteHBjM1FpT2xzel
↪10mkjiaaHuO4EdHXAxT6P-Q__f4ztOGgBNPsCIjFdf0
```

And the response:

```
HTTP/1.1 200 OK
Cache-Control: no-cache, private
Content-Type: application/json
Date: Thu, 22 Apr 2021 13:30:52 GMT, Thu, 22 Apr 2021 13:30:52 GMT
Host: 127.0.0.1:8000
X-Debug-Token: 47d353
X-Debug-Token-Link: http://127.0.0.1:8000/_profiler/47d353
X-Powered-By: PHP/7.4.16
X-Robots-Tag: noindex
Content-Length: 71
Connection: close

{
"key": "value",
"list": [
    1,
    2,
    3
],
"sub": "user_60817a5b312bd",
"active": true
}
```

CHAPTER

FIVE

TESTS, CODE QUALITY AND CODE STYLE

5.1 Local development

Follow the procedure described in the *Configuration* page to setup a local development environment.

5.2 Workflow

Every time changes are introduced into the library, Github Actions run the tests written with Behat.

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.

To run the whole tests tasks locally, do

```
./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 project.

```
./vendor/bin/grumphp run
GrumPHP is sniffing your code!
Running task  1/13: License... ✓
Running task  2/13: composer_require_checker... ✓
Running task  3/13: composer... ✓
Running task  4/13: ComposerNormalize... ✓
Running task  5/13: YamlLint... ✓
Running task  6/13: JsonLint... ✓
Running task  7/13: PhpLint... ✓
Running task  8/13: TwigCs... ✓
Running task  9/13: PhpCsFixer... ✓
Running task 10/13: Phpcs... ✓
Running task 11/13: PhpStan... ✓
Running task 12/13: Psalm... ✓
Running task 13/13: Behat... ✓
```

CHAPTER

SIX

CONTRIBUTING

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

CHAPTER

SEVEN

DEVELOPMENT

7.1 Maintainers

See the MAINTAINERS.txt file.

7.2 Documentation

Documentation can be built locally using Sphinx.

To render the documentation locally do the following steps:

- `docker-compose up`
- Navigate to http://127.0.0.1:8100/

7.3 Contributors

See the Github insights page.