PHP authentication bundles

for Symfony

DIGIT PHP Support - Pol Dellaiera
European Commission
December 2022

PHP authentication bundles

1980-01-01

PHP authentication bundles

for Symfony

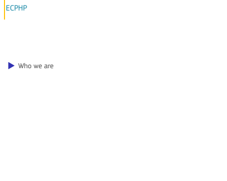
DIGIT PHP Support - Pol Dellaiera
European Commission
December 2022

Introduction

ECPHP

▶ Who we are

1980-01-01



ECPHP

▶ Who we are

How about starting with an introduction of the ECPHP team?

ECPHP

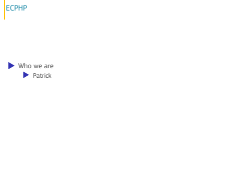
- Who we are
  - Patrick

1980-01-01

PHP authentication bundles

Introduction

ECPHP



ECPHP

- Who we are
  - Patrick

Patrick is managing the team and prioritizing the tasks to do

ECPHP

- Who we are
  - Patrick
  - Manos

ECPHP

- Who we are
  - Patrick
  - Manos

Emmanuel, the Patrick's backup

ECPHP

- Who we are
  - Patrick
  - Manos
  - Pol

ECPHP

- Who we are
  - Patrick
  - Manos
  - Pol

And then myself, I'm the developer of the team, while I'm mostly assisting developers with their app developments and corporate libraries development. I'm also experiment open source tools to ease the job of the developers, such tools as Nix, LaTeX, Pandoc.

ECPHP

- Who we are
  - Patrick
  - Manos
  - Pol
- What we do

1980-01-01

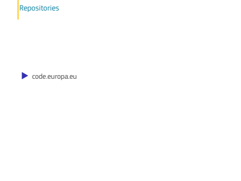
PHP authentication bundles
└ Introduction
  └ ECPHP

ECPHP

- Who we are
  - Patrick
  - Manos
  - Pol
- What we do

Repositories

- code.europa.eu



Repositories

code.europa.eu

We have the brand new code.europa.eu which is the main repository

Repositories

- code.europa.eu
- github.com

1980-01-01

Repositories

- code.europa.eu
- github.com

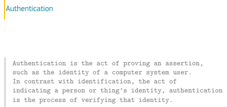
And we use Github as a backup repository

Authentication

European
Commission

Authentication

> Authentication is the act of proving an assertion,
> such as the identity of a computer system user.
> In contrast with identification, the act of
> indicating a person or thing’s identity, authentication
> is the process of verifying that identity.


Authentication

Authentication is the act of proving an assertion,
such as the identity of a computer system user.
In contrast with identification, the act of
indicating a person or thing’s identity, authentication
is the process of verifying that identity.

Let’s first start with a definition to make sure that everyone is on the same wavelength.

Protocols

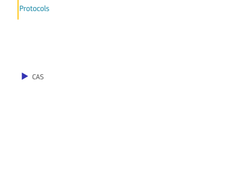
- CAS

1980-01-01

PHP authentication bundles

Authentication

Protocols


Protocols

- CAS

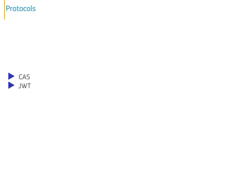
When we think about authentication, we think about security, protocols, single sign on. Protocols like...

- CAS for Central Authentication System which are redirection based protocols. They are using URL and redirections in your browser to authenticate users. CAS is used to authorize users to access an application, no users details are usually retrieved through it. - JWT for Json Web Token protocols like OAuth, OpenID Connect, etc etc , used for authorization but also for exchanging information since using JWT is a good way to securely transmitting information between parties

Protocols

- CAS
- JWT

European Commission


Protocols

- CAS
- JWT

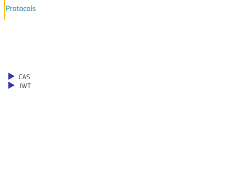
These protocols can be used to secure web applications, and rest APIs.

Protocols

- CAS
- JWT

1980-01-01

PHP authentication bundles
└ Authentication
  └ Protocols



Protocols

- CAS
- JWT

Implementing the security in an application is something that might quickly become cumbersome and time consuming, especially if the libraries to implement the protocols are not available.

Protocols

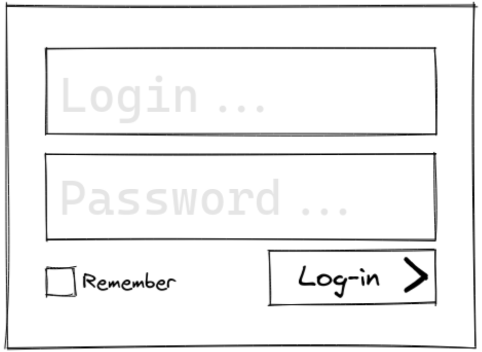
- CAS
- JWT

1980-01-01

PHP authentication bundles
- Authentication
  - Protocols

Luckily for you, those libraries are already existing in the PHP world and we published some of them in our team, in open-source.

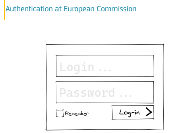
Authentication at European Commission



European Commission

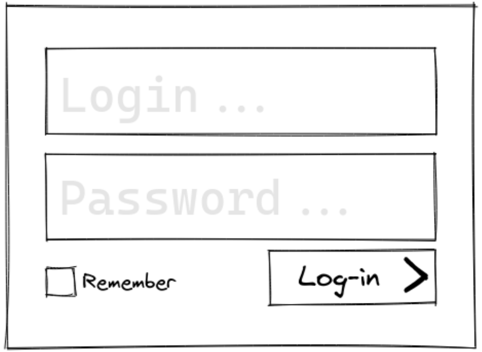
1980-01-01

PHP authentication bundles
- Authentication
  - Authentication at European Commission

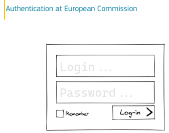

Authentication at European Commission

I guess you all know this when it comes to authentication.

Authentication at European Commission

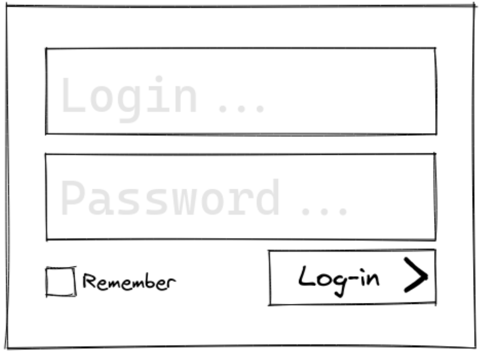


1980-01-01


Authentication at European Commission

In the EC context, when it comes to authentication, we often think to EU Login, the unique portal providing authentication for so many users, consultants and employees at EC.

Authentication at European Commission



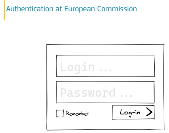
European Commission

PHP authentication bundles

Authentication

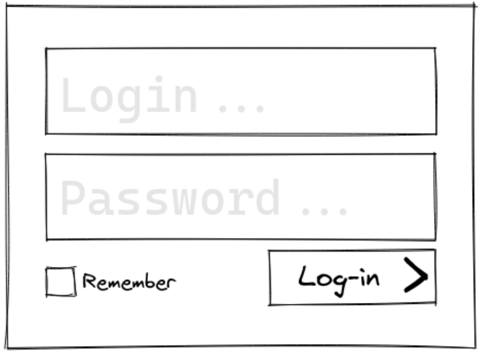
Authentication at European Commission

1980-01-01

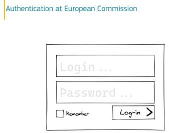


But have you ever think about the mechanism in-between your own application hosted on your own domain and EU Login, hosted on another domain.

Authentication at European Commission

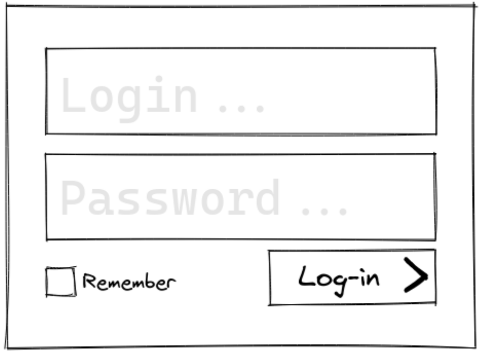


1980-01-01


Authentication at European Commission

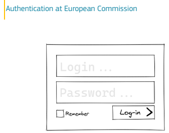
There are many ways to achieve that and today we are going to focus on the CAS protocol. The EU Login platform implements multiple authentication protocols and CAS is one of them.

Authentication at European Commission



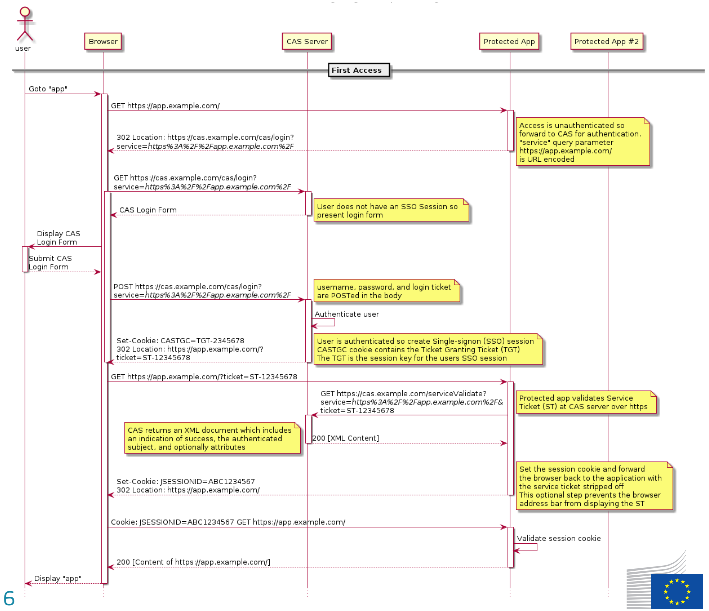
1980-01-01

PHP authentication bundles
└ Authentication
  └ Authentication at European Commission


Authentication at European Commission

Let’s unravel the mysteries of the CAS protocol… don’t be afraid!

CAS at European Commission



user

Browser

CAS Server

Protected App

Protected App #2

First Access

Goto "app"

GET https://app.example.com/

302 Location: https://cas.example.com/cas/login?
service=*https%3A%2F%2Fapp.example.com%2F*

Access is unauthenticated so
forward to CAS for authentication.
"service" query parameter
https://app.example.com/
is URL encoded

GET https://cas.example.com/cas/login?
service=*https%3A%2F%2Fapp.example.com%2F*

CAS Login Form

User does not have an SSO Session so
present login form

Display CAS
Login Form

Submit CAS
Login Form

POST https://cas.example.com/cas/login?
service=*https%3A%2F%2Fapp.example.com%2F*

username, password, and login ticket
are POSTed in the body

Authenticate user

Set-Cookie: CASTGC=TGT-2345678
302 Location: https://app.example.com/?
ticket=ST-12345678

User is authenticated so create Single-signon (SSO) session
CASTGC cookie contains the Ticket Granting Ticket (TGT)
The TGT is the session key for the users SSO session

GET https://app.example.com/?ticket=ST-12345678

GET https://cas.example.com/serviceValidate?
service=*https%3A%2F%2Fapp.example.com%2F&*
ticket=ST-12345678

Protected app validates Service
Ticket (ST) at CAS server over https

200 [XML Content]

CAS returns an XML document which includes
an indication of success, the authenticated
subject, and optionally attributes

Set-Cookie: JSESSIONID=ABC1234567
302 Location: https://app.example.com/

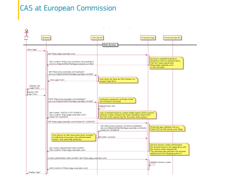
Set the session cookie and forward
the browser back to the application with
the service ticket stripped off
This optional step prevents the browser
address bar from displaying the ST

Cookie: JSESSIONID=ABC1234567 GET https://app.example.com/

Validate session cookie

200 [Content of https://app.example.com/]

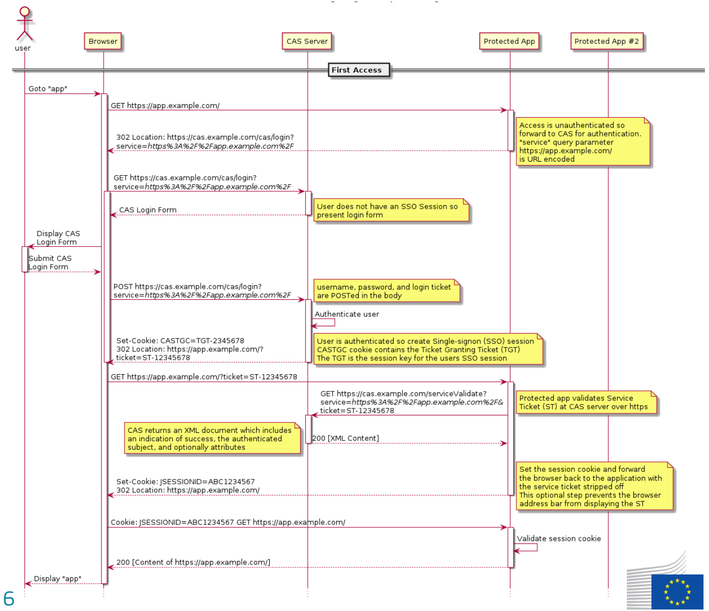
Display "app"



CAS at European Commission

Oops! Ok I guess you're afraid already! Sorry about that!

CAS at European Commission



user

Browser

CAS Server

Protected App

Protected App #2

First Access

Goto "app"

GET https://app.example.com/

Access is unauthenticated so forward to CAS for authentication. "service" query parameter https://app.example.com/ is URL encoded

302 Location: https://cas.example.com/cas/login?service=*https%3A%2F%2Fapp.example.com%2F*

GET https://cas.example.com/cas/login?service=*https%3A%2F%2Fapp.example.com%2F*

User does not have an SSO Session so present login form

CAS Login Form

Display CAS Login Form

Submit CAS Login Form

POST https://cas.example.com/cas/login?service=*https%3A%2F%2Fapp.example.com%2F*

username, password, and login ticket are POSTed in the body

Authenticate user

User is authenticated so create Single-signon (SSO) session CASTGC cookie contains the Ticket Granting Ticket (TGT) The TGT is the session key for the users SSO session

Set-Cookie: CASTGC=TGT-2345678
302 Location: https://app.example.com/?ticket=ST-12345678

GET https://app.example.com/?ticket=ST-12345678

GET https://cas.example.com/serviceValidate?service=*https%3A%2F%2Fapp.example.com%2F&*ticket=ST-12345678

Protected app validates Service Ticket (ST) at CAS server over https

CAS returns an XML document which includes an indication of success, the authenticated subject, and optionally attributes

200 [XML Content]

Set the session cookie and forward the browser back to the application with the service ticket stripped off This optional step prevents the browser address bar from displaying the ST

Set-Cookie: JSESSIONID=ABC1234567
302 Location: https://app.example.com/

Cookie: JSESSIONID=ABC1234567 GET https://app.example.com/

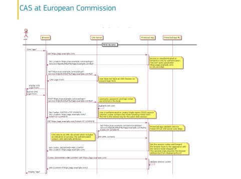
Validate session cookie

200 [Content of https://app.example.com/]

Display "app"

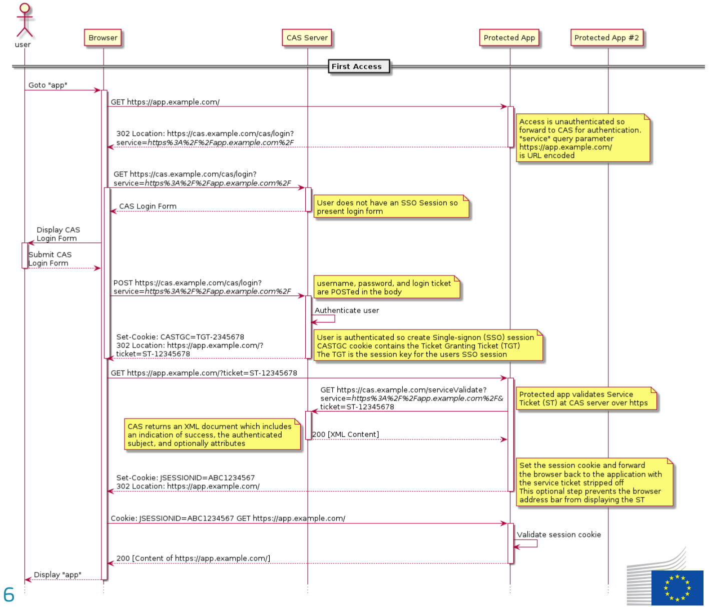
1980-01-01

PHP authentication bundles
└─Authentication
└─CAS at European Commission



Here’s the sequence diagram when you login through Eu Login using the CAS Protocol.

CAS at European Commission



user

Browser

CAS Server

Protected App

Protected App #2

First Access

Goto "app"

GET https://app.example.com/

302 Location: https://cas.example.com/cas/login?
service=*https%3A%2F%2Fapp.example.com%2F*

Access is unauthenticated so
forward to CAS for authentication.
"service" query parameter
https://app.example.com/
is URL encoded

GET https://cas.example.com/cas/login?
service=*https%3A%2F%2Fapp.example.com%2F*

CAS Login Form

User does not have an SSO Session so
present login form

Display CAS
Login Form

Submit CAS
Login Form

POST https://cas.example.com/cas/login?
service=*https%3A%2F%2Fapp.example.com%2F*

username, password, and login ticket
are POSTed in the body

Authenticate user

Set-Cookie: CASTGC=TGT-2345678
302 Location: https://app.example.com/?
ticket=ST-12345678

User is authenticated so create Single-signon (SSO) session
CASTGC cookie contains the Ticket Granting Ticket (TGT)
The TGT is the session key for the users SSO session

GET https://app.example.com/?ticket=ST-12345678

GET https://cas.example.com/serviceValidate?
service=*https%3A%2F%2Fapp.example.com%2F&*
ticket=ST-12345678

Protected app validates Service
Ticket (ST) at CAS server over https

200 [XML Content]

CAS returns an XML document which includes
an indication of success, the authenticated
subject, and optionally attributes

Set-Cookie: JSESSIONID=ABC1234567
302 Location: https://app.example.com/

Set the session cookie and forward
the browser back to the application with
the service ticket stripped off
This optional step prevents the browser
address bar from displaying the ST

Cookie: JSESSIONID=ABC1234567 GET https://app.example.com/

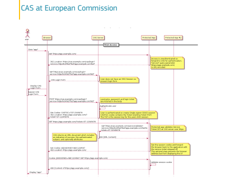
Validate session cookie

200 [Content of https://app.example.com/]

Display "app"

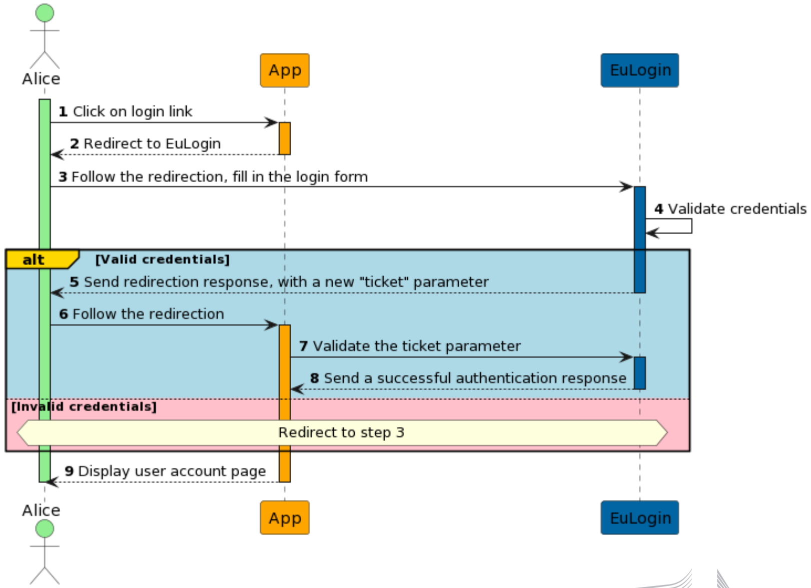
European
Commission

1980-01-01



But no worries, I made a simplified version of it...

CAS at European Commission



Alice

App

EuLogin

1 Click on login link

2 Redirect to EuLogin

3 Follow the redirection, fill in the login form

4 Validate credentials

alt **[Valid credentials]**

5 Send redirection response, with a new "ticket" parameter

6 Follow the redirection

7 Validate the ticket parameter

8 Send a successful authentication response

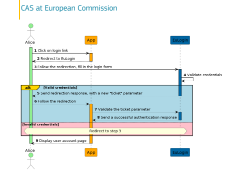
[Invalid credentials]

Redirect to step 3

9 Display user account page

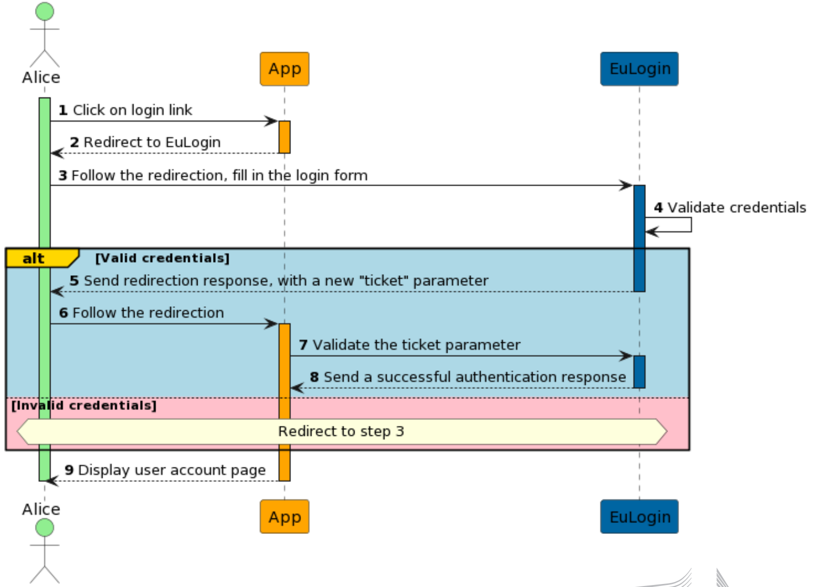
1980-01-01

PHP authentication bundles
└─Authentication
└─CAS at European Commission



In this sequence diagram, a user is trying to login onto a web app

CAS at European Commission



Alice

App

EuLogin

1 Click on login link

2 Redirect to EuLogin

3 Follow the redirection, fill in the login form

4 Validate credentials

alt **[Valid credentials]**

5 Send redirection response, with a new "ticket" parameter

6 Follow the redirection

7 Validate the ticket parameter

8 Send a successful authentication response

[Invalid credentials]

Redirect to step 3

9 Display user account page

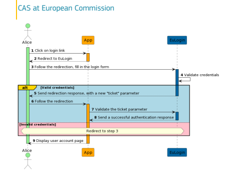
Alice

App

EuLogin

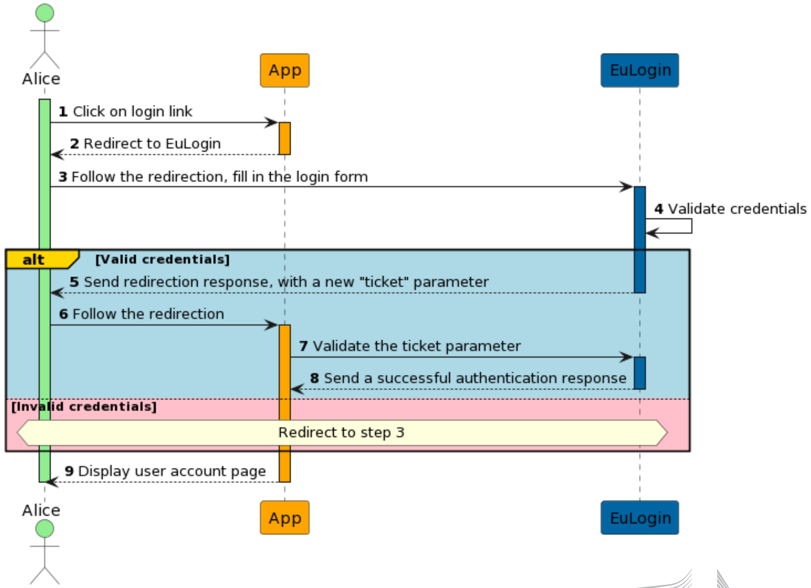
1980-01-01

PHP authentication bundles
└Authentication
└CAS at European Commission



At first the user open its browser, goes on the App and click the login button or link

CAS at European Commission



Alice

App

EuLogin

1 Click on login link

2 Redirect to EuLogin

3 Follow the redirection, fill in the login form

4 Validate credentials

alt **[Valid credentials]**

5 Send redirection response, with a new "ticket" parameter

6 Follow the redirection

7 Validate the ticket parameter

8 Send a successful authentication response

[Invalid credentials]

Redirect to step 3

9 Display user account page

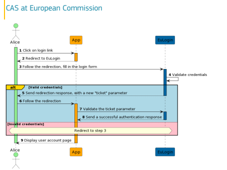
Alice

App

EuLogin

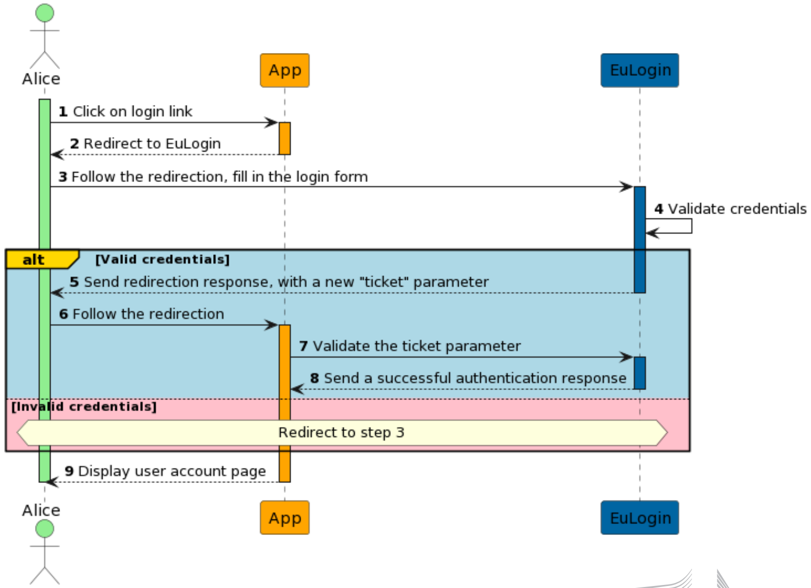
1980-01-01

PHP authentication bundles
└Authentication
└CAS at European Commission



Somehow, the App detects that the user is not authenticated and redirect the user to EU Login if so

CAS at European Commission



Alice — App — EuLogin

1 Click on login link

2 Redirect to EuLogin

3 Follow the redirection, fill in the login form

4 Validate credentials

alt [Valid credentials]

5 Send redirection response, with a new "ticket" parameter

6 Follow the redirection

7 Validate the ticket parameter

8 Send a successful authentication response

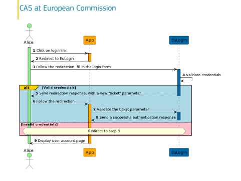
[Invalid credentials]

Redirect to step 3

9 Display user account page

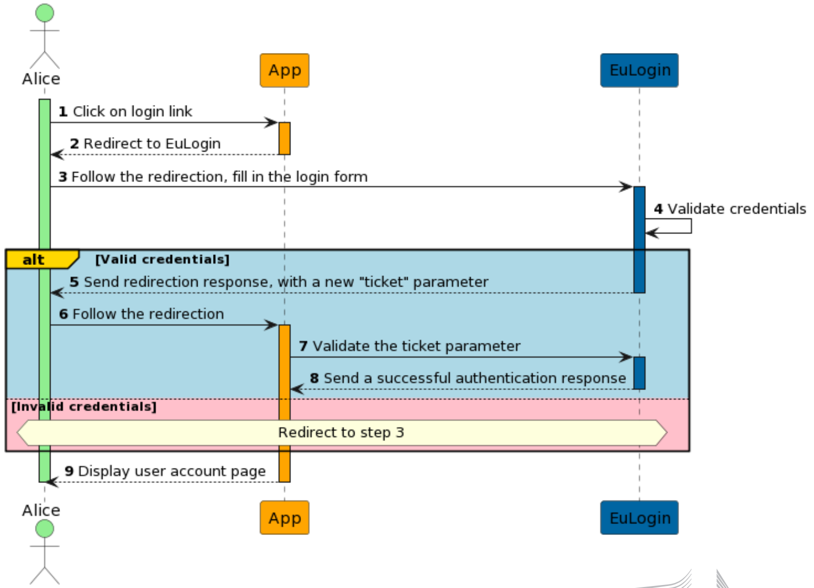
1980-01-01

PHP authentication bundles
└─Authentication
  └─CAS at European Commission



Eu Login display the login form and the user submit its credentials

CAS at European Commission



1 Click on login link

2 Redirect to EuLogin

3 Follow the redirection, fill in the login form

4 Validate credentials

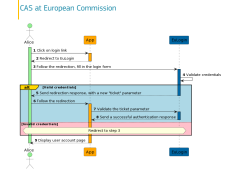
5 Send redirection response, with a new "ticket" parameter

6 Follow the redirection

7 Validate the ticket parameter

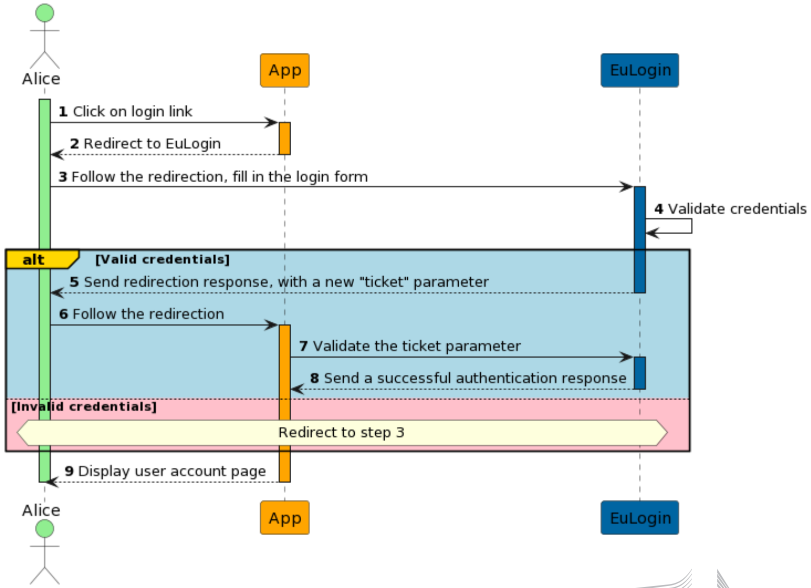
8 Send a successful authentication response

9 Display user account page



Eu Login redirects the user back to the application, appending a special ticket parameter in the url, depending on the validity of its credentials

CAS at European Commission



Alice

App

EuLogin

1 Click on login link

2 Redirect to EuLogin

3 Follow the redirection, fill in the login form

4 Validate credentials

alt **[Valid credentials]**

5 Send redirection response, with a new "ticket" parameter

6 Follow the redirection

7 Validate the ticket parameter

8 Send a successful authentication response

[Invalid credentials]

Redirect to step 3

9 Display user account page

Alice

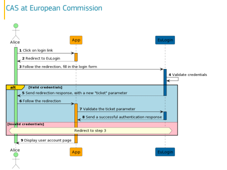
App

EuLogin

European Commission

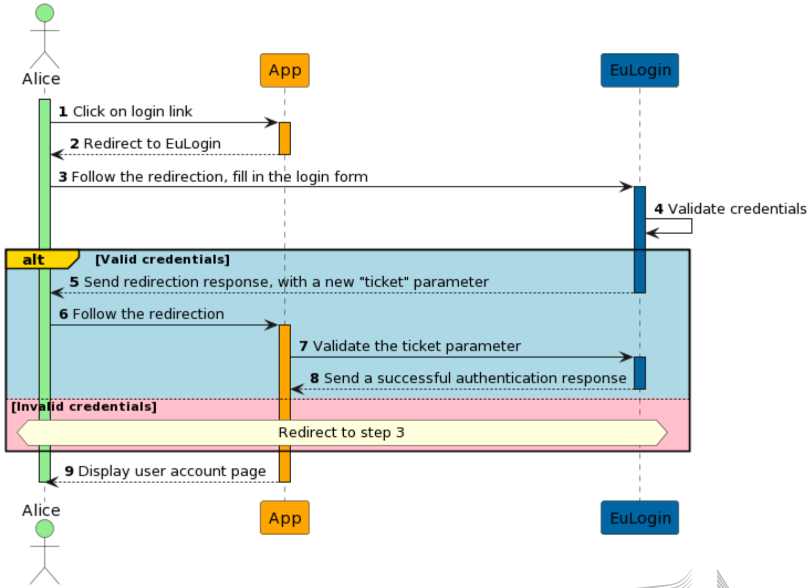
PHP authentication bundles
└ Authentication
  └ CAS at European Commission

1980-01-01



The application validate the ticket parameter in the URL, if any

CAS at European Commission



Alice

App

EuLogin

1 Click on login link

2 Redirect to EuLogin

3 Follow the redirection, fill in the login form

4 Validate credentials

alt **[Valid credentials]**

5 Send redirection response, with a new "ticket" parameter

6 Follow the redirection

7 Validate the ticket parameter

8 Send a successful authentication response

[Invalid credentials]

Redirect to step 3

9 Display user account page

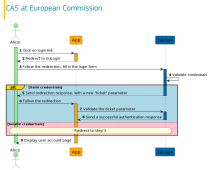
Alice

App

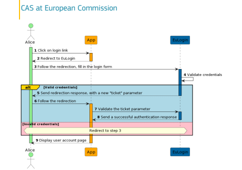
EuLogin

1980-01-01

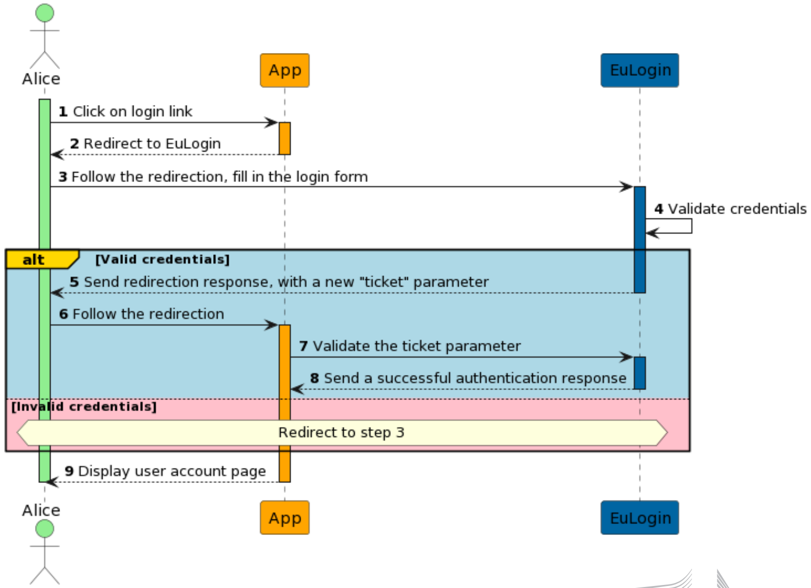
PHP authentication bundles
Authentication
CAS at European Commission



In case of valid credentials, the user is logged in or an error message is displayed



CAS at European Commission



1 Click on login link

2 Redirect to EuLogin

3 Follow the redirection, fill in the login form

4 Validate credentials

5 Send redirection response, with a new "ticket" parameter

6 Follow the redirection

7 Validate the ticket parameter

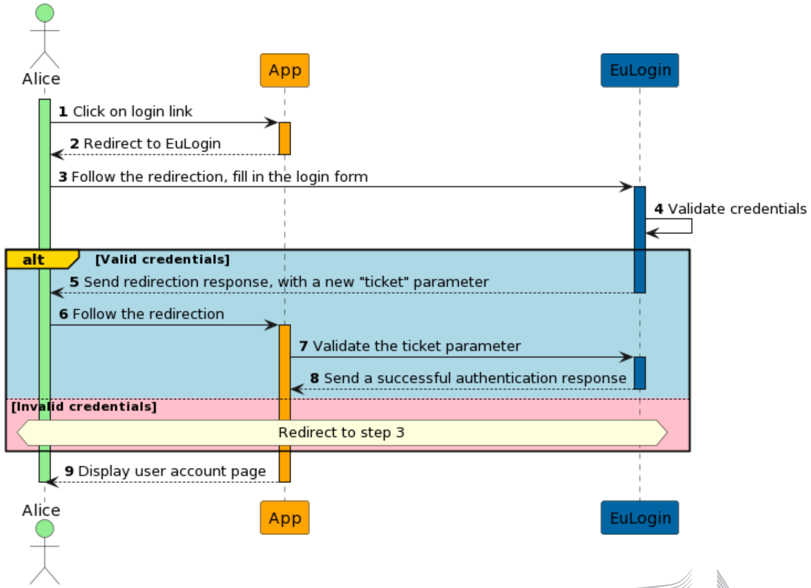
8 Send a successful authentication response

Redirect to step 3

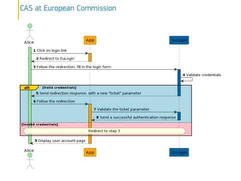
9 Display user account page

As I said, this is an extremely simplified diagram and there are more things that are done, but this is to give you an idea of what is going on when you login onto an app.

CAS at European Commission

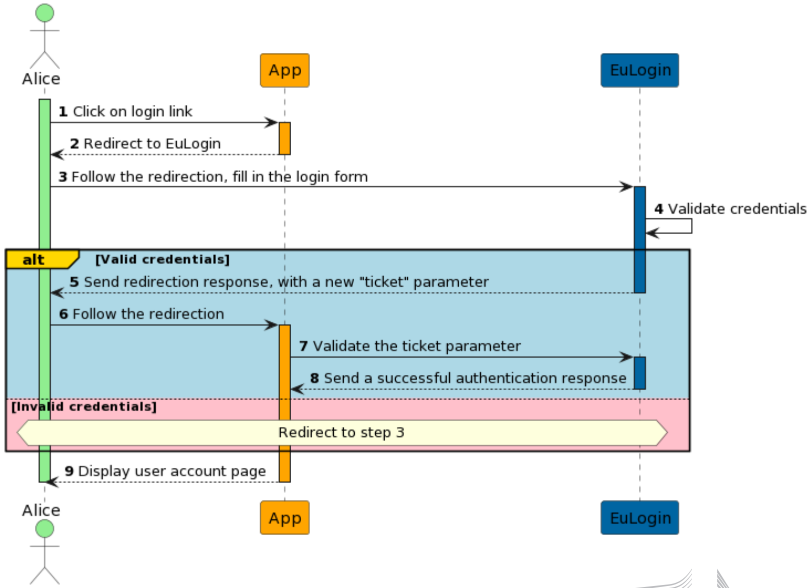

Alice
App
EuLogin
1 Click on login link
2 Redirect to EuLogin
3 Follow the redirection, fill in the login form
4 Validate credentials
alt
[Valid credentials]
5 Send redirection response, with a new "ticket" parameter
6 Follow the redirection
7 Validate the ticket parameter
8 Send a successful authentication response
[Invalid credentials]
Redirect to step 3
9 Display user account page
Alice
App
EuLogin

1980-01-01



We notice the interactions between the user and the App, the user and EuLogin and the App with EuLogin.

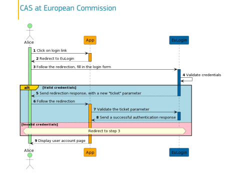
CAS at European Commission


Alice
App
EuLogin
1 Click on login link
2 Redirect to EuLogin
3 Follow the redirection, fill in the login form
4 Validate credentials
alt
[Valid credentials]
5 Send redirection response, with a new "ticket" parameter
6 Follow the redirection
7 Validate the ticket parameter
8 Send a successful authentication response
[Invalid credentials]
Redirect to step 3
9 Display user account page
Alice
App
EuLogin

European Commission

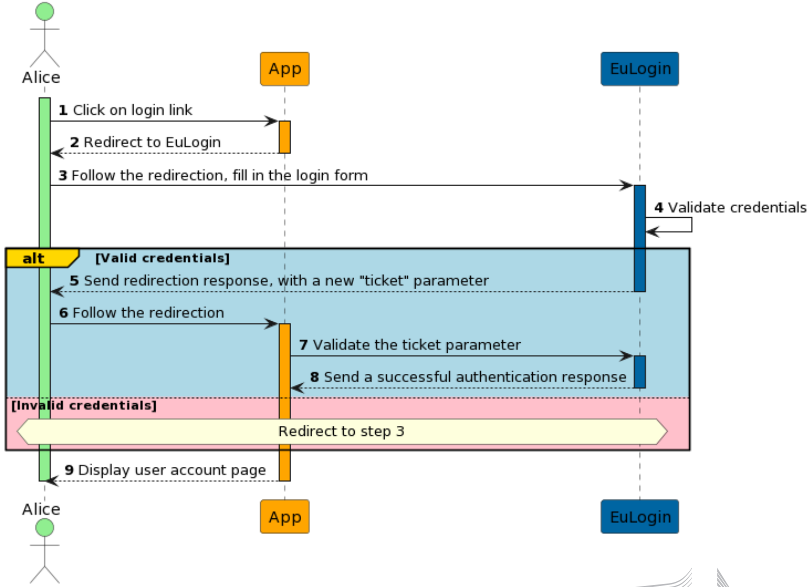
1980-01-01

PHP authentication bundles
└ Authentication
└ CAS at European Commission



If we only limit ourself to the CAS protocol for now, there are not a lot of options for the developers… CAS Lib can be used with or without a framework, DG Sante for example uses CAS Lib on a bare PHP app.

CAS at European Commission



Alice

App

EuLogin

1 Click on login link

2 Redirect to EuLogin

3 Follow the redirection, fill in the login form

4 Validate credentials

alt **[Valid credentials]**

5 Send redirection response, with a new "ticket" parameter

6 Follow the redirection

7 Validate the ticket parameter

8 Send a successful authentication response

[Invalid credentials]

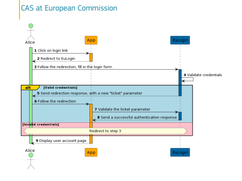
Redirect to step 3

9 Display user account page

Alice

App

EuLogin



This is how CAS-Lib was born… but before going on to that subject, let’s have a look at what we have in store…

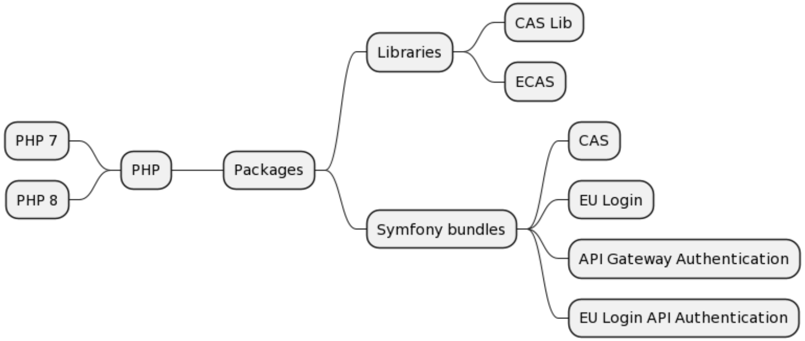
PHP

European
Commission

1980-01-01

PHP authentication bundles
└─ PHP

PHP libraries / bundles / packages



PHP Authentication libraries at ECPHP

PHP 7

PHP 8

PHP

Packages

Libraries

CAS Lib

ECAS

Symfony bundles

CAS

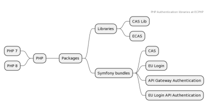
EU Login

API Gateway Authentication

EU Login API Authentication

1980-01-01

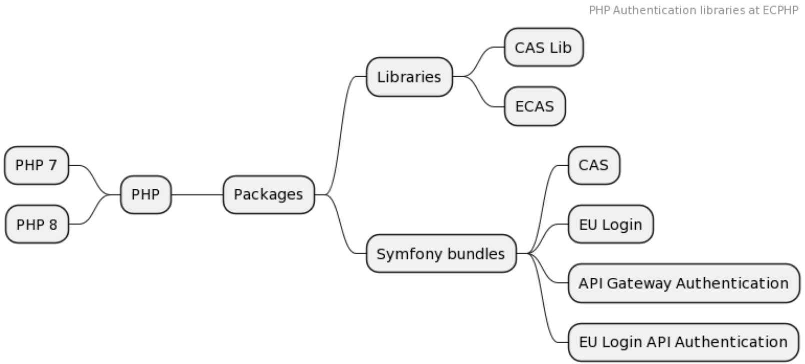
PHP authentication bundles
└PHP
└PHP libraries / bundles / packages



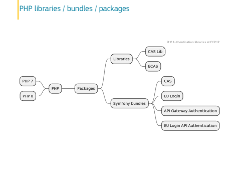
PHP libraries / bundles / packages

In this mindmap, you have a quick and global overview of the authentication packages that we provides at ECPHP.

PHP libraries / bundles / packages

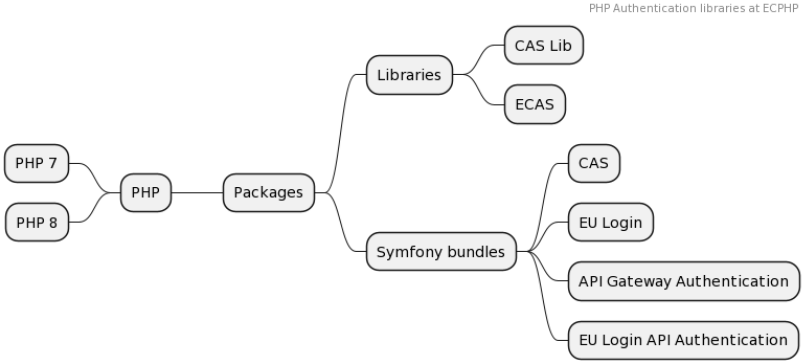

PHP Authentication libraries at ECPHP

1980-01-01


PHP libraries / bundles / packages

These packages are available from PHP 7 to 8 and we are slowly going to deprecate support for PHP 7

PHP libraries / bundles / packages

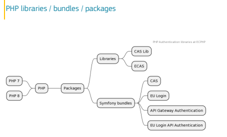


PHP Authentication libraries at ECPHP

PHP 7
PHP 8
PHP
Packages
Libraries
CAS Lib
ECAS
Symfony bundles
CAS
EU Login
API Gateway Authentication
EU Login API Authentication

European Commission

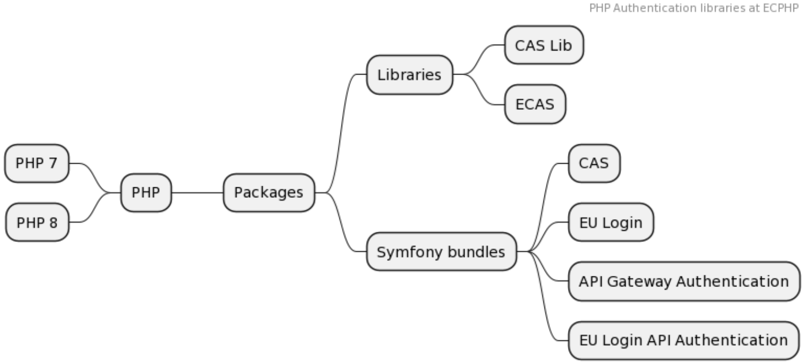
PHP authentication bundles
└PHP
└PHP libraries / bundles / packages



The first one we did was CAS Lib. A library to facilitate the communication with a standard CAS server. This is a library that I made on my own and moved to ECPHP. The work on this library started around November 2019. Initially called ”PCAS”, then renamed to ”PSRCAS” and finally to ”CAS Lib”.
That package is a standard PHP library that can be used by any PHP application, with or without a framework behind. It provides ways to authenticate a user session using the CAS protocol.
The CAS Protocol is not complex, it just boils down to sending HTTP responses, making requests, altering URLs, parsing JSON and XML, managing configuration... But just that in PHP can be a problem... <sarcasm>anyway, what is not a problem in PHP?</sarcasm>
- Sending HTTP redirections? Ok, but depending on where it is used, there might be plenty of different ways to send them.
- Altering URL? Seems to be the easiest part... erm wait, are we sure?
- Parsing JSON? Oh thatś actually the easiest, there are core functions in PHP to encode and decode json... but bummer, ‘json_decode‘ returns null for an invalid input... even though null is also a perfectly valid object for JSON to decode to! At the time of writing, it has been somehow fixed since PHP 7.3. Indeed, we can now throw an exception in case of issue while encoding or decoding a JSON string, but we have to add an optional flag while it should be the default behavior... Trust me, you’re going to love PHP if you don’t already.
- Parsing XML? Erm... Let me lol.
- Managing configuration? Seems easy at first sight.
In order to make it as standard as possible, it exclusively uses PSR interfaces, hence the name PSR-CAS ! Don’t worry, there’s a slide about PSR. Therefore, CAS Lib can be used in a Symfony project, Laravel project or any other framework.

A lot of effort into this library to make it consistent and well tested.

PHP libraries / bundles / packages



PHP Authentication libraries at ECPHP

PHP 7

PHP 8

PHP

Packages

Libraries

CAS Lib

ECAS

Symfony bundles

CAS

EU Login

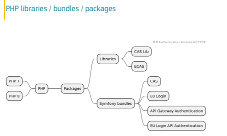
API Gateway Authentication

EU Login API Authentication

European Commission

1980-01-01

PHP authentication bundles
└ PHP
└ PHP libraries / bundles / packages

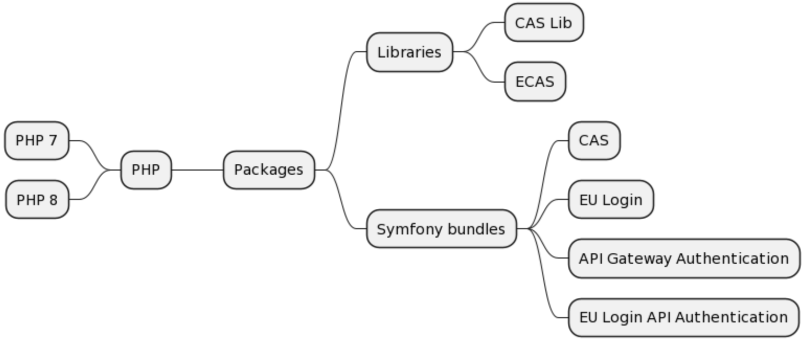


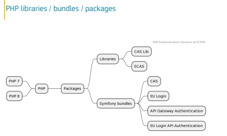
PHP libraries / bundles / packages

Then there is eCAS, a library decorating the CAS library and adding compatibility with the customizations made by European Commission to the CAS protocol.

PHP libraries / bundles / packages

PHP Authentication libraries at ECPHP

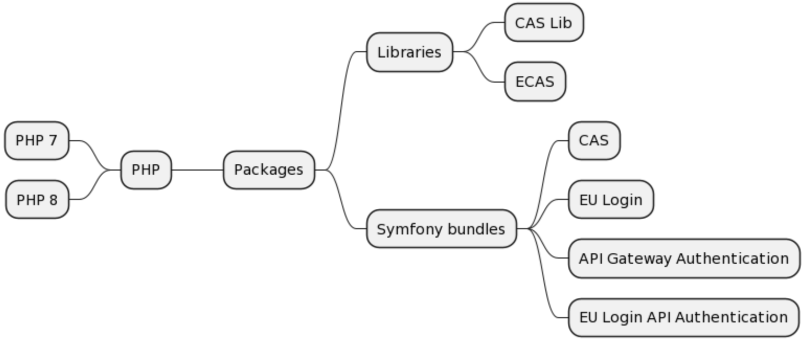



PHP libraries / bundles / packages

There are only 2 libraries in ECPHP, the rest are Symfony bundle, let's check them out...

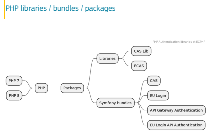
PHP libraries / bundles / packages

PHP Authentication libraries at ECPHP



1980-01-01

PHP authentication bundles
└─PHP
└─PHP libraries / bundles / packages

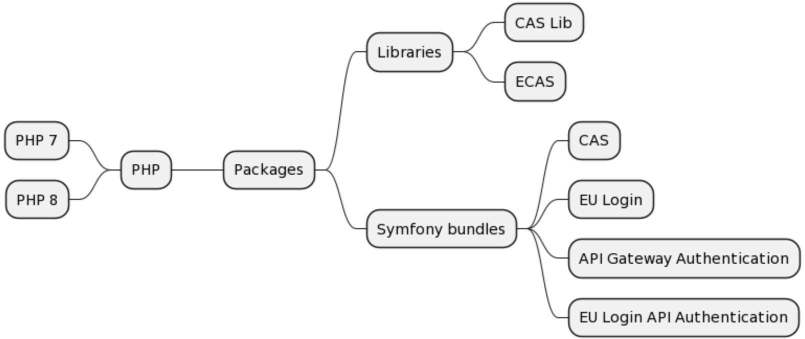


CAS-Bundle which is a bundle letting any Symfony application authenticate their users through the standard CAS protocol.

Basically, this is the glue code between Symfony and CAS-Lib.

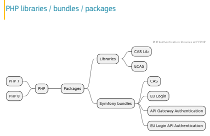
PHP libraries / bundles / packages

PHP Authentication libraries at ECPHP


PHP 7
PHP 8
PHP
Packages
Libraries
CAS Lib
ECAS
Symfony bundles
CAS
EU Login
API Gateway Authentication
EU Login API Authentication

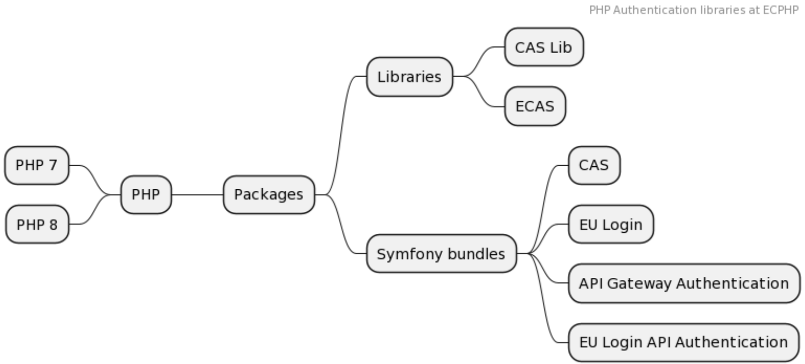
1980-01-01

PHP authentication bundles
└PHP
└PHP libraries / bundles / packages

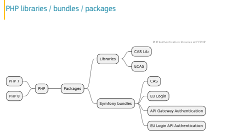


Then there is the EU-Login-Bundle counterpart, which is the same as CAS-Bundle, but for eCAS.

PHP libraries / bundles / packages

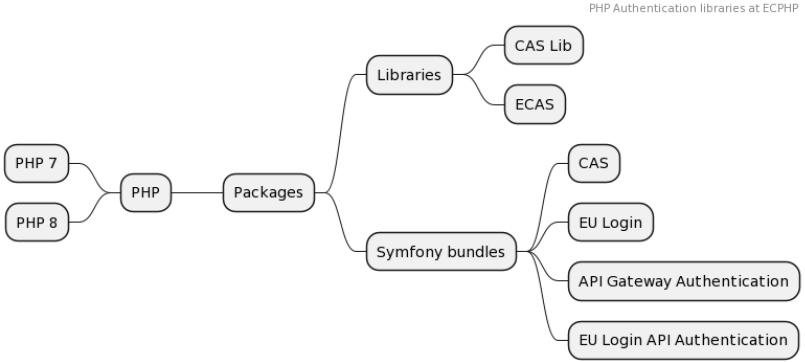

PHP Authentication libraries at ECPHP
PHP 7
PHP 8
PHP
Packages
Libraries
CAS Lib
ECAS
Symfony bundles
CAS
EU Login
API Gateway Authentication
EU Login API Authentication

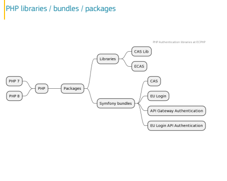
European Commission



Then, API-GW-Authentication, which is basically a JWT authentication with a mechanism to dynamically retrieve the JWKS keys from API Gateway.

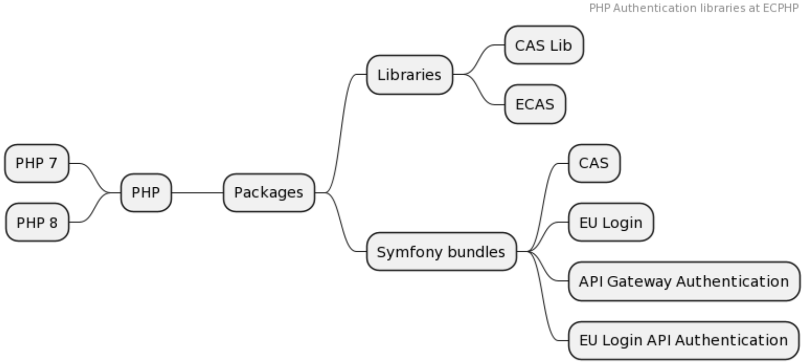
PHP libraries / bundles / packages

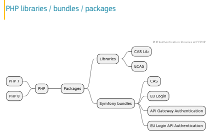

PHP Authentication libraries at ECPHP
PHP 7
PHP 8
PHP
Packages
Libraries
CAS Lib
ECAS
Symfony bundles
CAS
EU Login
API Gateway Authentication
EU Login API Authentication



And the last bundle, which was the most complicated to do: EU Login API authentication which uses OpenID Connect protocol to authenticate users.

PHP libraries / bundles / packages


PHP Authentication libraries at ECPHP
PHP 7
PHP 8
PHP
Packages
Libraries
CAS Lib
ECAS
Symfony bundles
CAS
EU Login
API Gateway Authentication
EU Login API Authentication



We made other bundles, but they are not relevant in the context of authentication.

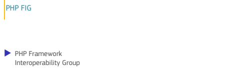
PHP FIG

- PHP Framework Interoperability Group

European
Commission

1980-01-01

PHP authentication bundles
- PHP
  - PHP FIG



PHP FIG

PHP Framework
Interoperability Group

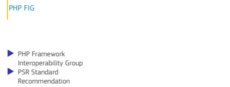
PHP FIG stands for PHP Framework Interoperability Group

PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation

1980-01-01

PHP authentication bundles
- PHP
  - PHP FIG


PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation

While PSR stands for PSR Standard Recommendation
The idea behind the FIG group is for project representatives to talk about the commonalities between our projects and find ways we can work together. If other folks want to adopt what they're doing they are welcome to do so, but that is not the aim. Nobody in the group wants to tell you, as a programmer, how to build your application.

They are responsible for the creation of the following PSRs

PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation
- PSR-0

1980-01-01

PHP authentication bundles
- PHP
  - PHP FIG

PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation
- PSR-0

In 2010, PSR-0, followed by PSR-4 in 2013 for providing the autoloading... this is thanks to those PSR that we have composer!

And without Composer... we would be nowhere.

PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation

- PSR-0
- PSR-3

1980-01-01

PHP authentication bundles
└ PHP
  └ PHP FIG

PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation
- PSR-0
- PSR-3

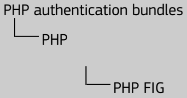
PSR-3 for providing a logger mechanism interface

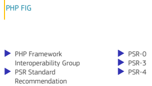
PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation

- PSR-0
- PSR-3
- PSR-4

1980-01-01




PHP FIG

PHP Framework Interoperability Group
PSR Standard Recommendation

PSR-0
PSR-3
PSR-4

In 2013, PSR-0 has been deprecated in favor of PSR-4

PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation

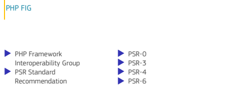
- PSR-0
- PSR-3
- PSR-4
- PSR-6

1980-01-01

PHP authentication bundles

PHP

PHP FIG



PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation

- PSR-0
- PSR-3
- PSR-4
- PSR-6

PSR-6 for providing Caching interfaces

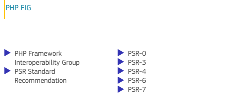
PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation

- PSR-0
- PSR-3
- PSR-4
- PSR-6
- PSR-7

1980-01-01

PHP authentication bundles
└ PHP
  └ PHP FIG


PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation

- PSR-0
- PSR-3
- PSR-4
- PSR-6
- PSR-7

PSR-7 for providing the Response and Request messages interface

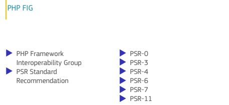
PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation

- PSR-0
- PSR-3
- PSR-4
- PSR-6
- PSR-7
- PSR-11

1980-01-01

PHP authentication bundles
└─PHP
　└─PHP FIG


PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation

- PSR-0
- PSR-3
- PSR-4
- PSR-6
- PSR-7
- PSR-11

PSR-11 for providing the container interface

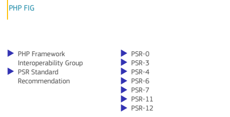
PHP FIG

- ▶ PHP Framework Interoperability Group
- ▶ PSR Standard Recommendation

- ▶ PSR-0
- ▶ PSR-3
- ▶ PSR-4
- ▶ PSR-6
- ▶ PSR-7
- ▶ PSR-11
- ▶ PSR-12

1980-01-01

PHP authentication bundles
└ PHP
  └ PHP FIG


PHP FIG

▶ PHP Framework Interoperability Group
▶ PSR Standard Recommendation

▶ PSR-0
▶ PSR-3
▶ PSR-4
▶ PSR-6
▶ PSR-7
▶ PSR-11
▶ PSR-12

PSR-12 for providing the coding standard rules

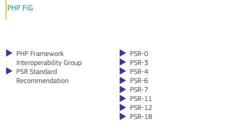
PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation

- PSR-0
- PSR-3
- PSR-4
- PSR-6
- PSR-7
- PSR-11
- PSR-12
- PSR-18

1980-01-01

- PHP authentication bundles
  - PHP
    - PHP FIG



PHP FIG

PHP Framework Interoperability Group
PSR Standard Recommendation

PSR-0
PSR-3
PSR-4
PSR-6
PSR-7
PSR-11
PSR-12
PSR-18

PSR 18 for providing the HTTP client interface

PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation

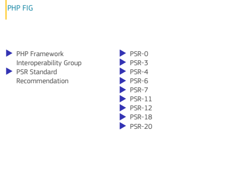
- PSR-0
- PSR-3
- PSR-4
- PSR-6
- PSR-7
- PSR-11
- PSR-12
- PSR-18
- PSR-20

1980-01-01

PHP authentication bundles

└ PHP

└ PHP FIG



PSR-20 for providing a Clock interface… work in ongoing…

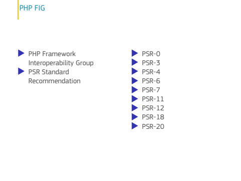
PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation

- PSR-0
- PSR-3
- PSR-4
- PSR-6
- PSR-7
- PSR-11
- PSR-12
- PSR-18
- PSR-20

1980-01-01

PHP authentication bundles
└PHP
  └PHP FIG


PHP FIG

- PHP Framework Interoperability Group
- PSR Standard Recommendation

- PSR-0
- PSR-3
- PSR-4
- PSR-6
- PSR-7
- PSR-11
- PSR-12
- PSR-18
- PSR-20

There are many other PSRs, I invite you to check them out by yourself

PHP FIG

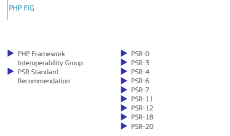
- PHP Framework Interoperability Group
- PSR Standard Recommendation

- PSR-0
- PSR-3
- PSR-4
- PSR-6
- PSR-7
- PSR-11
- PSR-12
- PSR-18
- PSR-20

European Commission

1980-01-01

PHP authentication bundles
└ PHP
  └ PHP FIG



PHP FIG

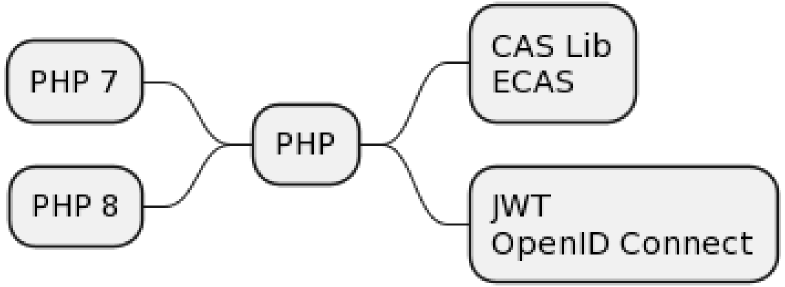
- PHP Framework Interoperability Group
- PSR Standard Recommendation

- PSR-0
- PSR-3
- PSR-4
- PSR-6
- PSR-7
- PSR-11
- PSR-12
- PSR-18
- PSR-20

All these PSRs are mostly providing interfaces (except PSR12) and you can use them anywhere. Thanks to the Liskov principle (1988) you are free to use any contrib implementations as long as they are implementing the proper required interface, you're good to go.

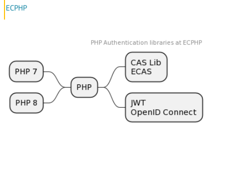
ECPHP

PHP Authentication libraries at ECPHP



1980-01-01

PHP authentication bundles
└ PHP
  └ ECPHP


ECPHP

PHP Authentication libraries at ECPHP

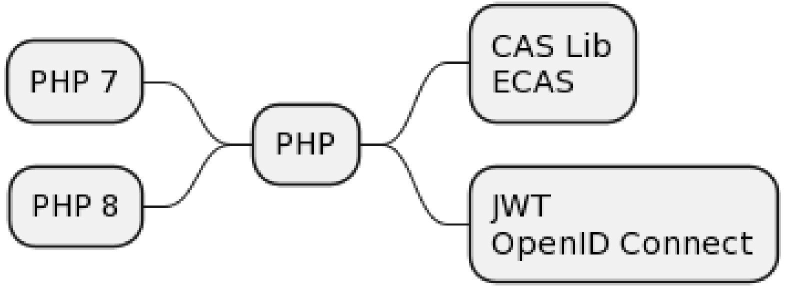
PHP 7
PHP 8
PHP
CAS Lib
ECAS
JWT
OpenID Connect

Let's come back to the main subject of this talk, authentication and summarize the protocols we made and which corresponding

PHP library package can be used. We have on the left the different authentication protocols that clients are using and…

ECPHP

PHP Authentication libraries at ECPHP


PHP 7

PHP 8

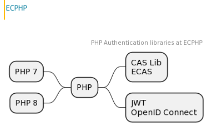
PHP

CAS Lib
ECAS

JWT
OpenID Connect

1980-01-01

PHP authentication bundles
- PHP
  - ECPHP


ECPHP

PHP Authentication libraries at ECPHP

PHP 7

PHP 8

PHP

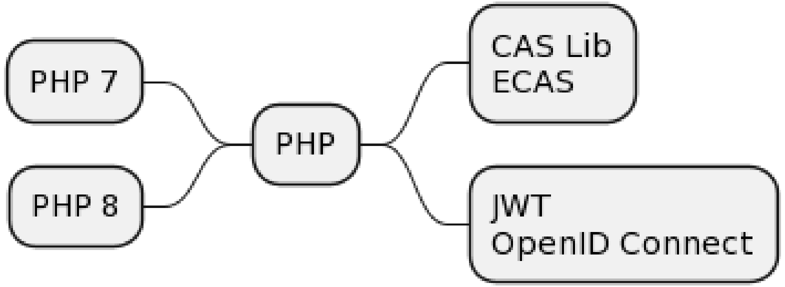
CAS Lib
ECAS

JWT
OpenID Connect

On the right, we have the corresponding packages in PHP

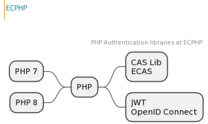
ECPHP

PHP Authentication libraries at ECPHP



1980-01-01

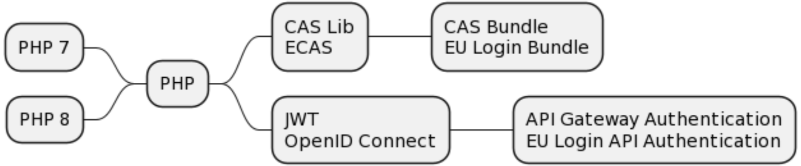
- PHP authentication bundles
  - PHP
    - ECPHP



The CAS protocol in CAS-Lib and CAS-Bundle

ECPHP

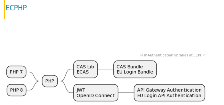
PHP Authentication libraries at ECPHP



European Commission

1980-01-01

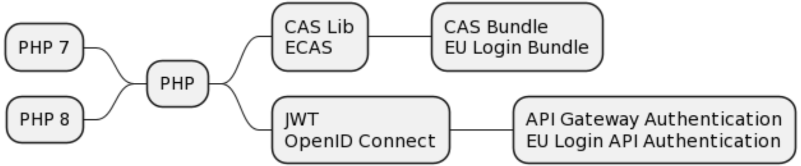
PHP authentication bundles
└─PHP
  └─ECPHP



The ECAS protocol in ECAS and EU-Login-Bundle

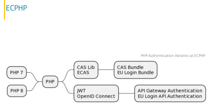
ECPHP

PHP Authentication libraries at ECPHP



1980-01-01

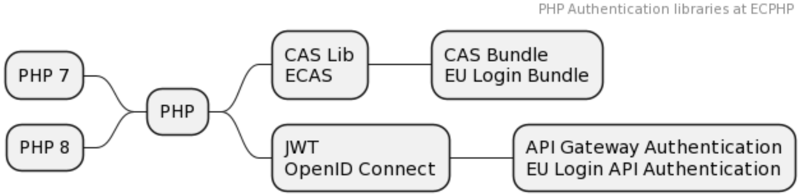
PHP authentication bundles
└─PHP
  └─ECPHP


ECPHP

PHP Authentication libraries at ECPHP

The Token authentication in API GW Authentication bundle, based on a popular existing contrib bundle

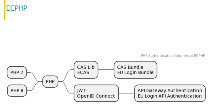
ECPHP


PHP Authentication libraries at ECPHP
PHP 7
PHP 8
PHP
CAS Lib
ECAS
CAS Bundle
EU Login Bundle
JWT
OpenID Connect
API Gateway Authentication
EU Login API Authentication

European Commission

1980-01-01

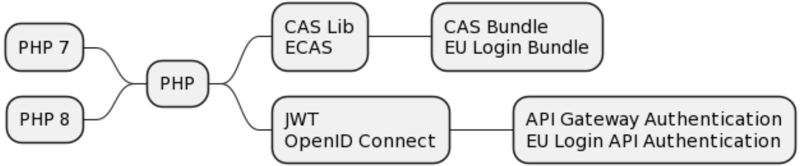
PHP authentication bundles
└─PHP
└─ECPHP



And the last one, The OpenID Connect authentication in EU Login API Authentication bundle, based on an existing OpenID Connect contrib library.

ECPHP

PHP Authentication libraries at ECPHP



PHP 7

PHP 8

PHP

CAS Lib
ECAS

CAS Bundle
EU Login Bundle

JWT
OpenID Connect

API Gateway Authentication
EU Login API Authentication

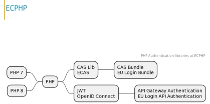
European
Commission

1980-01-01

PHP authentication bundles

└ PHP

└ ECPHP



ECPHP

PHP Authentication libraries at ECPHP

PHP 7

PHP 8

PHP

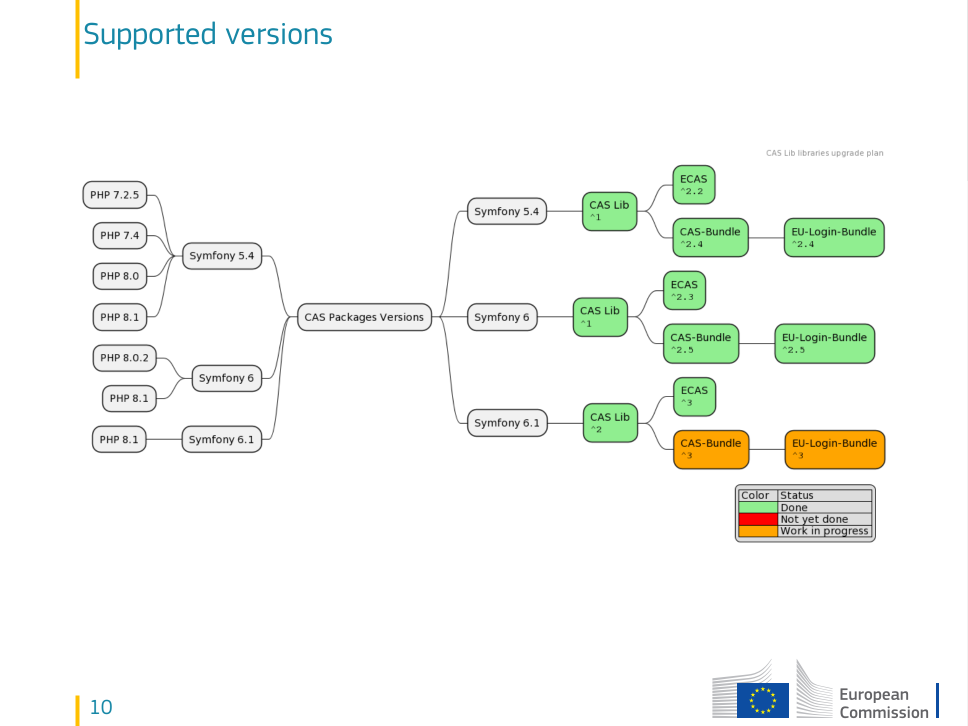
CAS Lib
ECAS

CAS Bundle
EU Login Bundle

JWT
OpenID Connect

API Gateway Authentication
EU Login API Authentication

Now you may wonder, is there a package for my own version of Symfony?



Supported versions

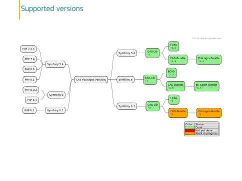
CAS Lib libraries upgrade plan

Color	Status
🟩	Done
🟥	Not yet done
🟧	Work in progress

European Commission

1980-01-01

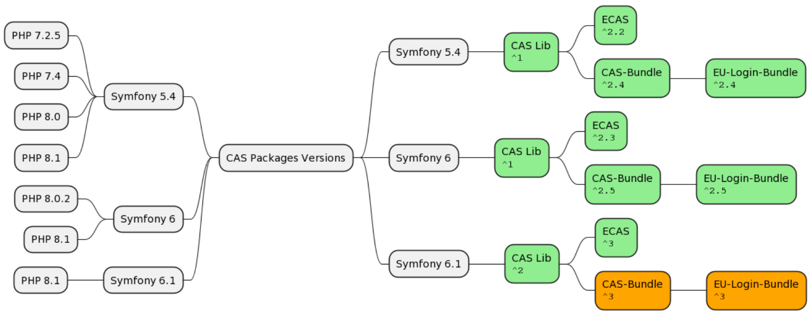
PHP authentication bundles
└PHP
└Supported versions



In this graphic, where only the CAS related packages are displayed, you have some kind of matrix of supported versions we are maintaining.

Supported versions

CAS Lib libraries upgrade plan

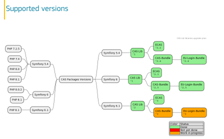


PHP 7.2.5
PHP 7.4
PHP 8.0
PHP 8.1
Symfony 5.4
PHP 8.0.2
PHP 8.1
Symfony 6
PHP 8.1
Symfony 6.1
CAS Packages Versions
Symfony 5.4
CAS Lib ^1
ECAS ^2.2
CAS-Bundle ^2.4
EU-Login-Bundle ^2.4
Symfony 6
CAS Lib ^1
ECAS ^2.3
CAS-Bundle ^2.5
EU-Login-Bundle ^2.5
Symfony 6.1
CAS Lib ^2
ECAS ^3
CAS-Bundle ^3
EU-Login-Bundle ^3

Color	Status
🟩	Done
🟥	Not yet done
🟧	Work in progress

1980-01-01

PHP authentication bundles
└ PHP
└ Supported versions

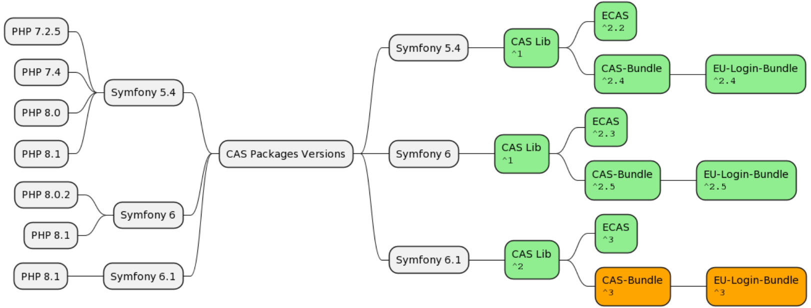


Supported versions

Almost all the Symfony versions are covered, from 4.4 to 6.2 which are not on this map. Symfony 6.2 has been released last week, on the 29 of November by the way. If you're late to the party, it's still time to upgrade your app :)

Supported versions

CAS Lib libraries upgrade plan



PHP 7.2.5
PHP 7.4
PHP 8.0
PHP 8.1
Symfony 5.4

PHP 8.0.2
PHP 8.1
Symfony 6

PHP 8.1
Symfony 6.1

CAS Packages Versions

Symfony 5.4
CAS Lib ^1
ECAS ^2.2
CAS-Bundle ^2.4
EU-Login-Bundle ^2.4

Symfony 6
CAS Lib ^1
ECAS ^2.3
CAS-Bundle ^2.5
EU-Login-Bundle ^2.5

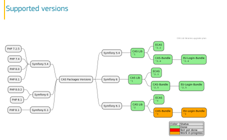
Symfony 6.1
CAS Lib ^2
ECAS ^3
CAS-Bundle ^3
EU-Login-Bundle ^3

Color	Status
🟩	Done
🟥	Not yet done
🟧	Work in progress

European Commission

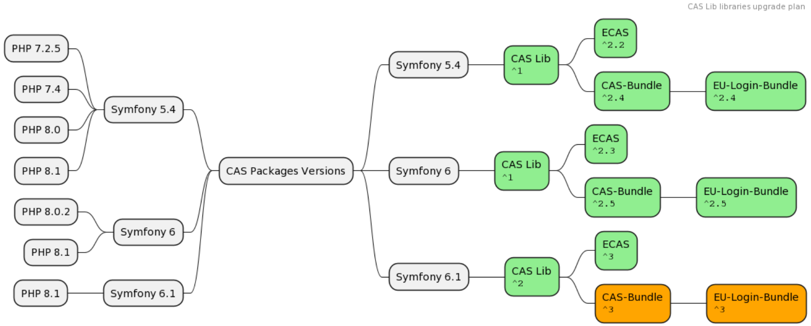
1980-01-01

PHP authentication bundles
└ PHP
└ Supported versions



There are still two bundles in orange because I'm currently having multiple issues with one of them, and I'm unable to make an short reproducer.

Supported versions


CAS Lib libraries upgrade plan

PHP 7.2.5
PHP 7.4
PHP 8.0
PHP 8.1
Symfony 5.4

PHP 8.0.2
PHP 8.1
Symfony 6

PHP 8.1
Symfony 6.1

CAS Packages Versions

Symfony 5.4
CAS Lib ^1
ECAS ^2.2
CAS-Bundle ^2.4
EU-Login-Bundle ^2.4

Symfony 6
CAS Lib ^1
ECAS ^2.3
CAS-Bundle ^2.5
EU-Login-Bundle ^2.5

Symfony 6.1
CAS Lib ^2
ECAS ^3
CAS-Bundle ^3
EU-Login-Bundle ^3

Color	Status
🟩	Done
🟥	Not yet done
🟧	Work in progress

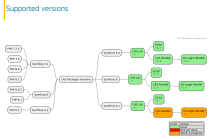
European Commission

1980-01-01

PHP authentication bundles

PHP

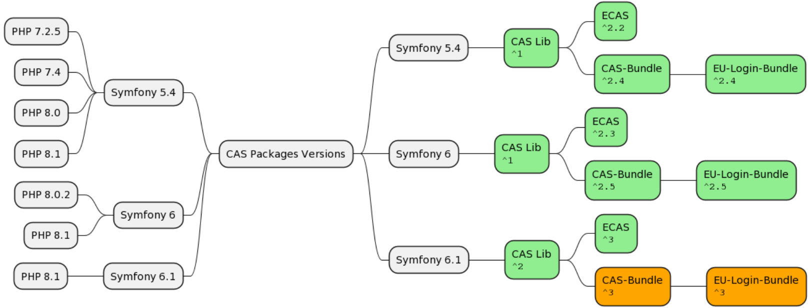
Supported versions



One of the issue has been reported and fixed in Symfony (https://github.com/symfony/symfony/pull/47808) but the other one is still open. If you want to help, let me know!

Supported versions

CAS Lib libraries upgrade plan



PHP 7.2.5

PHP 7.4

PHP 8.0

PHP 8.1

Symfony 5.4

PHP 8.0.2

PHP 8.1

Symfony 6

PHP 8.1

Symfony 6.1

CAS Packages Versions

Symfony 5.4

CAS Lib ^1

ECAS ^2.2

CAS-Bundle ^2.4

EU-Login-Bundle ^2.4

Symfony 6

CAS Lib ^1

ECAS ^2.3

CAS-Bundle ^2.5

EU-Login-Bundle ^2.5

Symfony 6.1

CAS Lib ^2

ECAS ^3

CAS-Bundle ^3

EU-Login-Bundle ^3

Color	Status
🟩	Done
🟥	Not yet done
🟧	Work in progress

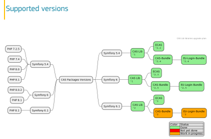
European Commission

1980-01-01

PHP authentication bundles

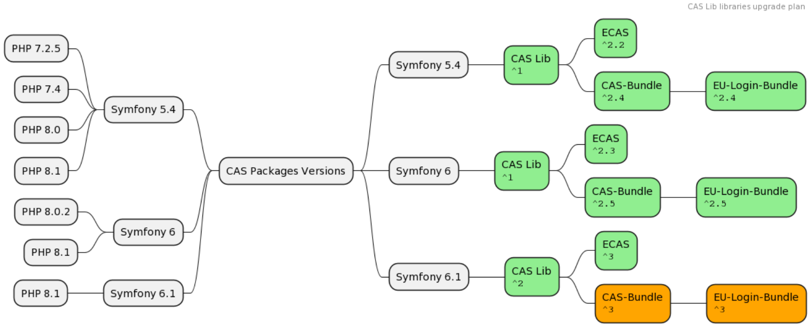
PHP

Supported versions



You may also notice that there is a major version jump for CAS Lib between Symfony 6 and 6.1.

Supported versions



CAS Lib libraries upgrade plan

CAS Packages Versions

PHP 7.2.5, PHP 7.4, PHP 8.0, PHP 8.1 → Symfony 5.4

PHP 8.0.2, PHP 8.1 → Symfony 6

PHP 8.1 → Symfony 6.1

Symfony 5.4 → CAS Lib ^1 → ECAS ^2.2; CAS-Bundle ^2.4 → EU-Login-Bundle ^2.4

Symfony 6 → CAS Lib ^1 → ECAS ^2.3; CAS-Bundle ^2.5 → EU-Login-Bundle ^2.5

Symfony 6.1 → CAS Lib ^2 → ECAS ^3; CAS-Bundle ^3 → EU-Login-Bundle ^3

Color	Status
🟩	Done
🟥	Not yet done
🟧	Work in progress

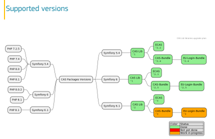
European Commission

1980-01-01

PHP authentication bundles

PHP

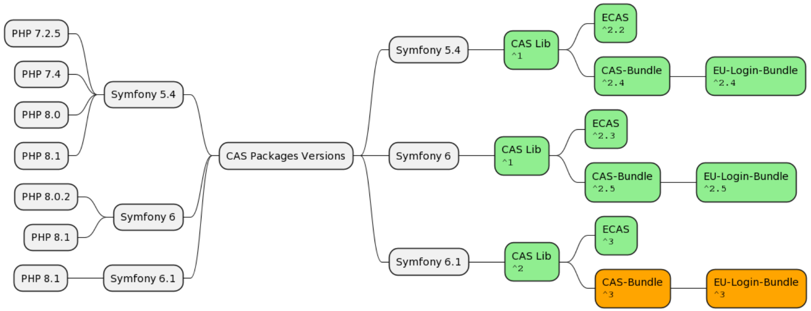
Supported versions



The reason is simple, CAS Lib has been completely rewritten recently, getting rid of the ghosts from the past...

Supported versions

CAS Lib libraries upgrade plan

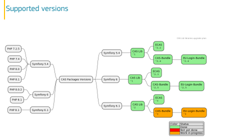


Color	Status
🟩	Done
🟥	Not yet done
🟧	Work in progress

European Commission

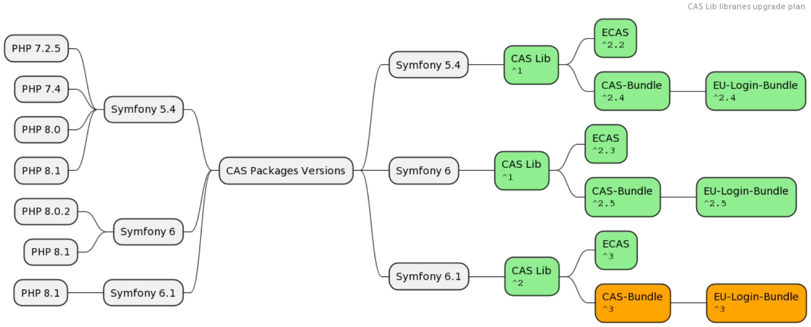
1980-01-01

PHP authentication bundles
└ PHP
└ Supported versions

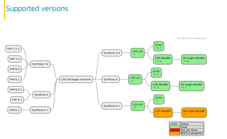


In ECPHP, we love the SOLID principles, do you know them?
The SOLID ideas are:
- The Single-responsibility principle: "There should never be more than one reason for a class to change.". In other words, every class should have only one responsibility.
- The Open–closed principle: "Software entities should be open for extension, but closed for modification."
- The Liskov substitution principle: "Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it.". In other words, use interfaces!
- The Interface segregation principle: "Clients should not be forced to depend upon interfaces that they do not use."
- The Dependency inversion principle: "Depend upon abstractions and not concretions."
If you really are interested, we can do a small session about it... but only if you really are interested! :)

Supported versions



Color	Status
🟩	Done
🟥	Not yet done
🟧	Work in progress

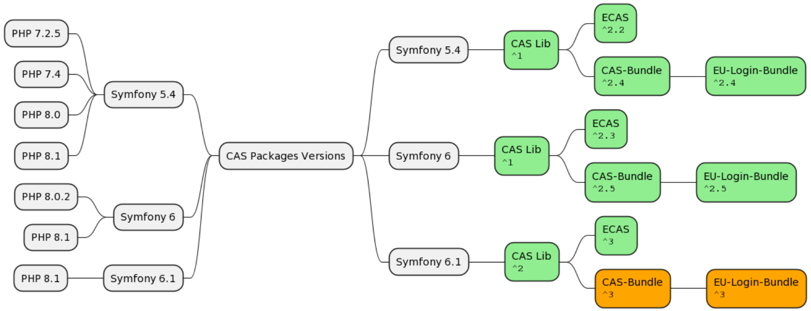

Supported versions

CAS Lib has been completely rewritten for many reasons:
- The main CAS service was stateful. The CAS service was holding input data in some class properties. Basically the Symfony request was injected in the constructor. This forced us to tweak the Symfony container to do so and it reduce the flexibility of the library and introduces bugs at some point, see (https://github.com/ecphp/cas-bundle/issues/63)

- The CAS service was mutable, because of the aforementioned reason

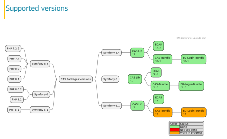
Supported versions

CAS Lib libraries upgrade plan



PHP 7.2.5
PHP 7.4
PHP 8.0
PHP 8.1
Symfony 5.4
PHP 8.0.2
PHP 8.1
Symfony 6
PHP 8.1
Symfony 6.1
CAS Packages Versions
Symfony 5.4
CAS Lib ^1
ECAS ^2.2
CAS-Bundle ^2.4
EU-Login-Bundle ^2.4
Symfony 6
CAS Lib ^1
ECAS ^2.3
CAS-Bundle ^2.5
EU-Login-Bundle ^2.5
Symfony 6.1
CAS Lib ^2
ECAS ^3
CAS-Bundle ^3
EU-Login-Bundle ^3

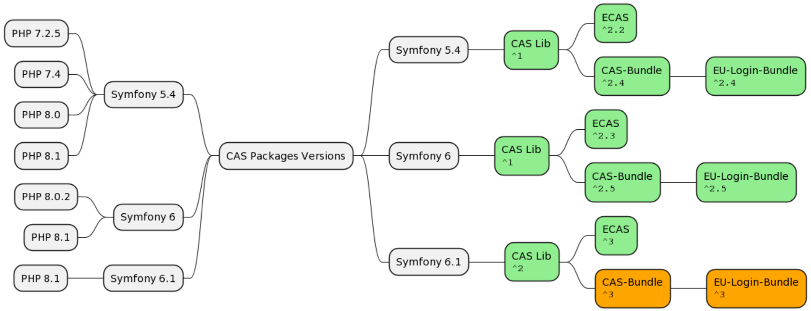
Color	Status
🟩	Done
🟥	Not yet done
🟧	Work in progress



There are still work to do in CAS Lib, like rewriting the tests using PHPUnit. Indeed, currently we use PHPSpec, but we would like to get rid of it everywhere for some reasons. It hasn't been done in version 2 because the amount of tests is actually greater than the library itself and it would have been too much at once. Also, rewriting the tests do not require a major version bump, so it can be done at anytime. Willing to help? Contact me!

Supported versions

CAS Lib libraries upgrade plan



PHP 7.2.5
PHP 7.4
PHP 8.0
PHP 8.1
Symfony 5.4

PHP 8.0.2
PHP 8.1
Symfony 6

PHP 8.1
Symfony 6.1

CAS Packages Versions

Symfony 5.4
CAS Lib ^1
ECAS ^2.2
CAS-Bundle ^2.4
EU-Login-Bundle ^2.4

Symfony 6
CAS Lib ^1
ECAS ^2.3
CAS-Bundle ^2.5
EU-Login-Bundle ^2.5

Symfony 6.1
CAS Lib ^2
ECAS ^3
CAS-Bundle ^3
EU-Login-Bundle ^3

Color	Status
🟩	Done
🟥	Not yet done
🟧	Work in progress

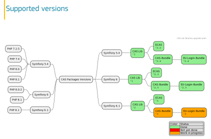
European Commission

PHP authentication bundles

PHP

Supported versions

1980-01-01

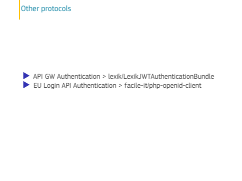


Right, I think we’ve said enough on the CAS protocol. What about the rest?

Other protocols

- API GW Authentication > lexik/LexikJWTAuthenticationBundle
- EU Login API Authentication > facile-it/php-openid-client

European
Commission


Other protocols

- API GW Authentication > lexik/LexikJWTAuthenticationBundle
- EU Login API Authentication > facile-it/php-openid-client

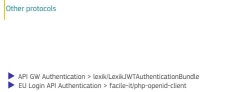
It is true that there won’t be so much details about other bundles because they basically rely on existing popular libraries that are working well since years.

Other protocols

- API GW Authentication > lexik/LexikJWTAuthenticationBundle
- EU Login API Authentication > facile-it/php-openid-client

1980-01-01

PHP authentication bundles
└ PHP
└ Other protocols


Other protocols

- API GW Authentication > lexik/LexikJWTAuthenticationBundle
- EU Login API Authentication > facile-it/php-openid-client

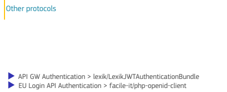
For API GW Authentication, it relies on LexikJWTAuthenticationBundle, the only added value of the bundle is a custom KeyLoader which automatically retrieve the JWKS keys automatically with a failsafe mechanism.

Other protocols

- API GW Authentication > lexik/LexikJWTAuthenticationBundle
- EU Login API Authentication > facile-it/php-openid-client

PHP authentication bundles
└ PHP
  └ Other protocols

1980-01-01


Other protocols

- API GW Authentication > lexik/LexikJWTAuthenticationBundle
- EU Login API Authentication > facile-it/php-openid-client

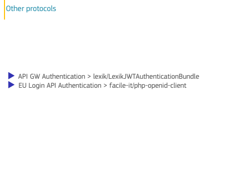
For EU Login API Authentication, we rely on facile-it/php-openid-client which does the heavy lifting job for us. Nevertheless, that bundle was the most complicated one to build.

Other protocols

- API GW Authentication > lexik/LexikJWTAuthenticationBundle
- EU Login API Authentication > facile-it/php-openid-client

1980-01-01

PHP authentication bundles
└PHP
  └Other protocols

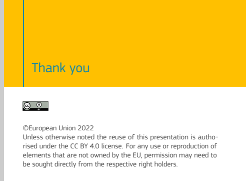


Other protocols

- API GW Authentication > lexik/LexikJWTAuthenticationBundle
- EU Login API Authentication > facile-it/php-openid-client

One of the challenge we had during its creation was the ability to generate fake valid token for the user so the application can be tested without relying on the internet.

Thank you



Thank you

©European Union 2022

Unless otherwise noted the reuse of this presentation is authorised under the CC BY 4.0 license. For any use or reproduction of elements that are not owned by the EU, permission may need to be sought directly from the respective right holders.