User's Guide for OpenSSO PHP Extension

Andreas Åkre Solberg

Pat Patterson

Table of Contents

Overview 1
Installation 1
Configuration 2
Meta Data 2
Setup SP meta data for Service Provider 3
Setp IdP meta data for Service Provider 3
Load SP meta data in your SAML 2.0 IdP 4
Plugins 5
Name Mapping Plugins 5
Session Management Plugins 6
Examples 6
Andreas' Example 6
Pat's Example 7

Overview

To OpenSSO PHP Extension up running, you need to perform the following steps:

1. Install the software, see the section called “Installation”
2. Configure the software, see the section called “Configuration”
3. Setup SP and IdP meta data for OpenSSO PHP Extension, see the section called “Meta Data”
4. Write and load SP meta data for the SAML 2.0 SP into your IdP, see the section called “Meta Data”

Installation

To install OpenSSO PHP Extension is as simple as just dropping the `openssophp` folder on a webaccessible area on your webserver. You are strongly advised to configure your web server to use HTTPS, because OpenSSO PHP Extension is using `Browser/POST` meaning that private data flows through HTTP. After you have installed the software, you should configure it.

Important

The instructions above will help you perform a basic installation. Some of the available plugins may require further steps. Please read the documentation of the combination of plugins you are using for more instructions in the section called “Plugins”. A basic installation will at least work with `namemapping/`

`transient` and `sessionmanagement/phpsession`.

Configuration

There is one configuration file and it is located in `openssophp/config/config.php`. Here is an example of a configuration:

```
$LIGHTBULB_CONFIG = array (
        'basedir'                                 => '/home/as/erlang/feide/openss
        'baseurl'                         => 'https://feide.erlang.no/openssophp/'

         'spi-sessionhandling'         => 'phpsession',
         'spi-namemapping'                 => 'database',

         'userdatabase'                  => array (
                 'username'                  => 'openssodemo',
                 'password'                  => 'abc123',
                 'host'                          => 'localhost',
                 'database'                  => 'openssodemo'
         ),

        'defaultLandingPage' => 'https://feide.erlang.no/example-andreas/test.php'
);
```

Here follows a description of each configuration entry:

basedir	This is the path on the server where the OpenSSO PHP Extension is installed. You may safely rename the openssophp folder to something different if you also update the path here in the configuration and in the meta data.
baseurl	This is the url where the `openssophp` folder is accessible from the web.
spi-sessionhandling	Type in the ID of the name mapping plugin that you want to use, see the section called “Session Management Plugins” for more information.
spi-namemapping	Type in the ID of the session handling plugin that you want to use, see the section called “Name Mapping Plugins” for more information.
userdatabase	This is the configuration of how to connect to the user database. This is only required for the `namemapping/database` plugin, see the section called “namemapping/database” for more information. If you are not using this plugin you can leave the default unchanged, or you can safely remote that section from the configuration file.
defaultLandingPage	If a SAML authentication response is received with no RelayState parameter, then the browser is redirected to this page.

Meta Data

To setup trust and relation between a OpenSSO PHP Extension SAML 2.0 SP and a SAML 2.0 IdP, the following steps needs to be performed:

1. Setup SP meta data for OpenSSO PHP Extension SAML 2.0 SP
2. Setup IdP meta data for OpenSSO PHP Extension SAML 2.0 SP
3. Create and load SP meta data for a SAML 2.0 IdP

To configure OpenSSO PHP Extension to work with mulitple IdPs, step 2 and 3 should be repeated.

Setup SP meta data for Service Provider

The meta data for SPs is located in `openssophp/config/saml-metadata-SP.php`. This file contains a list of SP meta data, each identified by a meta alias. For most use cases configuring one SP should be sufficient. The meta alias "`/sp`" can be left as it is. For each SP the following needs to be specified:

assertionConsumerServiceURL	The URL where the Browser/POST authentication response is sent. Make sure the hostname is correct.
issuer	This is the SAML 2.0 entity ID of the SP. If you unsure on what entity id to use, using the hostname of the SP is a good idea.
spNameQualifier	This value can be used to affiliate multiple SPs. If you are not using SAML 2.0 affiliations, you probably want to leave this field identical to the issuer.

Below is a sample meta data for one SP.

```
$spMetadata = array( "/sp" =>
  array( "assertionConsumerServiceURL"=>"https://feide.erlang.no/openssophp/As
         "issuer"=>"feide.erlang.no",
         "spNameQualifier" => "feide.erlang.no"));
```

Setp IdP meta data for Service Provider

Next, we want to configure the SP to talk to a SAML 2.0 IdP. The meta data is located in `openssophp/config/saml-metadata-IdP.php`. Below is a sample of three IdPs.

```
$idpMetadata = array(
        "sam.feide.no" =>
                    array( "SingleSignOnUrl"=>"https://sam.feide.no/amserver/S
         "SingleLogOutUrl"=>"https://sam.feide.no/amserver/IDPSloRedirect/meta
         "certFingerprint"=>"3a:e7:d3:d3:06:ba:57:fd:7f:62:6a:4b:a8:64:b3:4a:5

    "mars.feide.no" =>
                    array( "SingleSignOnUrl"=>"https://mars.feide.no/amserver/
         "SingleLogOutUrl"=>"https://mars.feide.no/amserver/IDPSloRedirect/met
         "certFingerprint"=>"d8:ee:63:c8:c3:0a:9d:f2:4b:7f:c2:7b:43:4c:85:8c:6
               "idp.simplesign.com" =>
                    array(
                     "SingleSignOnBinding" => "urn:oasis:names:tc:SAML:2.0:bin
                     "certificate" => "-----BEGIN CERTIFICATE-----\n"
                                                                 ."MIIhjkHDEJKjbjke
                                                                 ."FLEHOhohoehOEOBN
                                                                 ."...\n"
                                                                 ."FHJKEFJEbjkebjke
                                                                 ."-----END CERTIFI
);
```

Each IdP is identified by the SAML 2.0 Entity ID, and has a subset of the following properties associated with it:

SingleSignOnUrl	The URL to where the authentication request HTTP-REDIRECT is sent.
SingleLogOutUrl	The URL to where the logout request HTTP-REDIRECT is sent.
certFingerprint	The value of the fingerprint of the certificate the IdP is using to sign the assertion. The fingerprint should be in lowercase hex format, as seen in the example above. This property is currently only used with the "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" binding.
SingleSignOnBinding	The SAML 2.0 binding in use for single sign-on. If omitted, defaults to "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST". Only "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" and "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-SimpleSign" are currently supported.
certificate	The value of the certificate the IdP is using to sign the assertion. The fingerprint should be in Privacy-Enhanced Mail (PEM) format, as seen in the example above. This property is currently only used with the "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-SimpleSign" binding.

Load SP meta data in your SAML 2.0 IdP

First create a SAML 2.0 meta data document matching your OpenSSO PHP Extension SAML 2.0 SP. You may use the document below as a template:

```
<EntityDescriptor
    xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
    entityID="feide.erlang.no">
    <SPSSODescriptor
        AuthnRequestsSigned="false"
        WantAssertionsSigned="false"
        protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <SingleLogoutService
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
            Location="https://feide.erlang.no/openssophp/SingleLogoutService.php"
        <NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</NameIDF
        <NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</NameID
        <AssertionConsumerService isDefault="true" index="0"
            Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
            Location="https://feide.erlang.no/openssophp/AssertionConsumerService.
    </SPSSODescriptor>
</EntityDescriptor>
```

Remember to replace the hostnames of the SAML endpoints to match the host where you run the SP. You also must set the entityID.

Consult the documentation of your SAML 2.0 IdP software for how to load meta data for new service providers.

Example 1. Loading meta data in an OpenSSO IdP

To load meta data for a SP into an OpenSSO IdP, you can run a command similar to:

```
saml2meta import -u uid=amAdmin,ou=People,dc=feide,dc=no -w PASSWORD -m /opt/SUNWa
```

If you already have import that meta data and want to upload with changes, you first have to delete the entity, and then reload again with the command above. Here is a command to delete the entity:

```
saml2meta delete -u  uid=amAdmin,ou=People,dc=feide,dc=no -w PASSWORD -e feide.erl
```

Plugins

Currently OpenSSO PHP Extension has two plugin APIs. One API for custom name mapping and one for session management. At configuration time you will choose exactly one name mapping plugin and one session management plugin to be active, see more in the section called “Configuration”.

Name Mapping Plugins

The name mapping plugin is responsible for transforming a NameID to a local user ID. It has one function for setting the mapping, and one for looking up a mapping.

namemapping/transient

The `namemapping/transient` plugin is extremely simple. It requires no database, and does not really transform the NameID to a local ID, but just returns a transient NameID as the local ID. This plugin shuold be used together with transient NameIDMapping, but can also be used for persistent mapping. The typical use case for this plugin is where no user accounts exist in the service, but the service want to use attributes to perform access control. Another use case is a demo example, when you want to get OpenSSO PHP Extension up and running without the extra hassle with a database.

namemapping/database

The `namemapping/database` plugin will store and retrieve mappings from NameID to local IDs in a MySQL database. This database contains a table with all users in the system, and another table that contains the NameID mappings. The plugin it self will not do user administration for you, you will need to setup your user administration tool to work towards this database, or make changes to this plugin to access your existing userdatabase.

Before you can use this plugin, you must prepare the database. First, you should create a new MySQL database, and optionally a new database user. Next, you should initialize the database. A complete SQL initialization script is located under `etc/dbinit.sql`. This sript will create the tables and two users, `johns` and `admin`. Look in the script to see the local password. To execute the SQL script you will need to run a command similar to:

```
mysql -u openssodemo -p openssodemo < etc/dbinit.sql
```

After setting up a database, you have to configure access to the database in the OpenSSO PHP Extension config file (see the section called “Configuration”). This includes username, password, hostname and database.

If you want to understand this namemapping plugin better, you can access the mysql database, and query the content of the tables. Here are some examples:

To list all users:

```
mysql> select * from users;
+--------+----------+---------------+
| userid | password | username      |
+--------+----------+---------------+
| johns  | 123      | John Smith    |
| admin  | 456      | Administrator |
+--------+----------+---------------+
2 rows in set (0.01 sec)
```

To add a new user:

```
mysql> insert into users (userid,password,username) values ('andreas', 'foobar', '
Query OK, 1 row affected (0.01 sec)
```

To list all current namemappings:

```
mysql> select * from nameidmapping;
+---------------+-----------------+------------------------------+---------+
| idp           | sp              | nameid                       | localid |
+---------------+-----------------+------------------------------+---------+
| sam.feide.no  | feide.erlang.no | BfLY1GOZyc8dKE8AW5ELUH/EvfEZ | johns   |
| mars.feide.no | feide.erlang.no | LyDwR+FODi5sXwDYOIOAxw7R9ybl | johns   |
+---------------+-----------------+------------------------------+---------+
2 rows in set (0.01 sec)
```

To delete all namemappings. This will mean that all users will be defederated. Look at Pat's example (see the section called “Pat's Example”) for more information of how federating users works in real life.

```
mysql> delete from nameidmapping;
Query OK, 2 rows affected (0.02 sec)
```

Session Management Plugins

A session management plugin will create user sessions, usually by adding cookies to the user's browser, and allow the software to store and retrieve session attributes.

sessionmanagement/phpsession

The `sessionmanagement/phpsession` plugin reuses the built-in session management in PHP.

Examples

Currently two examples are provided with the OpenSSO PHP Extentsion package. You will find two folders, `example-andreas` and `example-pat`. These examples will help you verify your configuration, perform demonstrations, test your IdP, better understand OpenSSO PHP Extension and use as inspiration for how to integrate your service with OpenSSO PHP Extension.

Andreas' Example

This example is extremely simple to setup, and requires no databases.

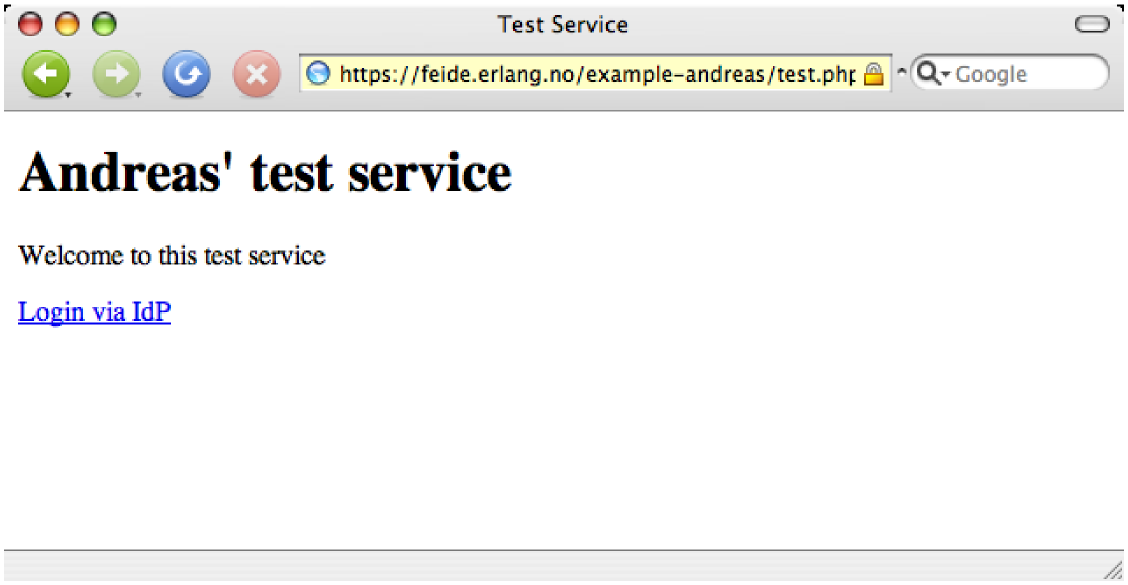
Start with editing the config file (see the section called “Configuration”) to use these two plugins: namemapping/transient and `sessionhandling/`phpsession. Next, configiure one SP and one IdP, as described in the section called “Meta Data”.

The example consist of one file only: `example-andreas/test.php`. You will not need to edit anything in that file.

Warning

If you have tested another example earlier and not properly logged out, you need to clear your cookies before testing this example.

Go to the URL pointing at the example, similar to `https://feide.erlang.no/example-andreas/test.php`, but at your server. You will then see that you are unauthenticated:

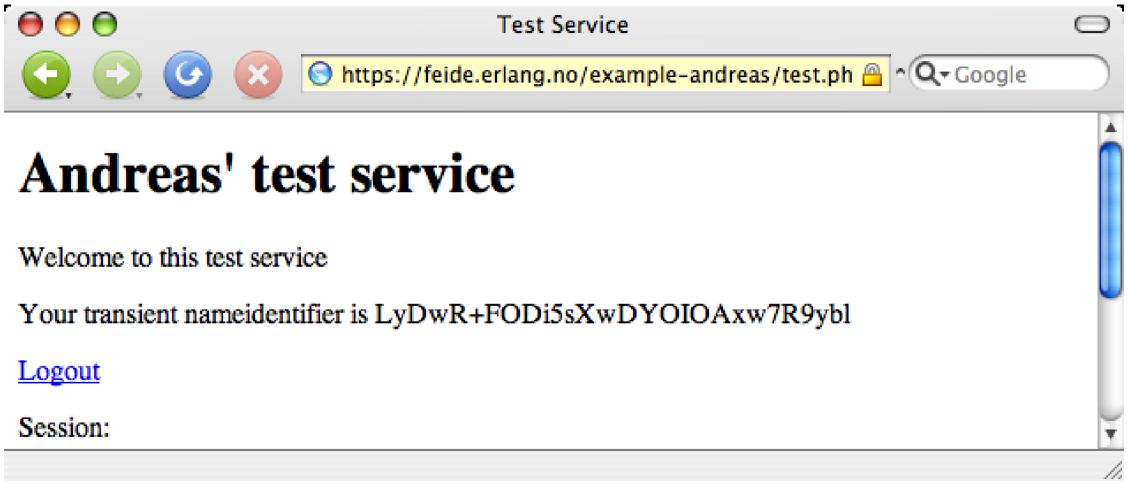


Andreas' test service

Welcome to this test service

Login via IdP

When you click on the link, you are redirected to your IdP, and you will be able to enter the credentials. Next, you should be redirected back to the example, and a page similar to this should show up:



Andreas' test service

Welcome to this test service

Your transient nameidentifier is LyDwR+FODi5sXwDYOIOAxw7R9ybl

Logout

Session:

The page shows your transient nameidentifier, which will not tell you anything, and some debug information. Hopefully your attribtues should also appear. You could use this example as a basis if you want a service that does not need to know the identity of the user, but instead need to perform access controll validation based on some of the user's attributes.

Pat's Example

This example is a bit more complex. It demonstrates federating local user accounts with a remote IdP.

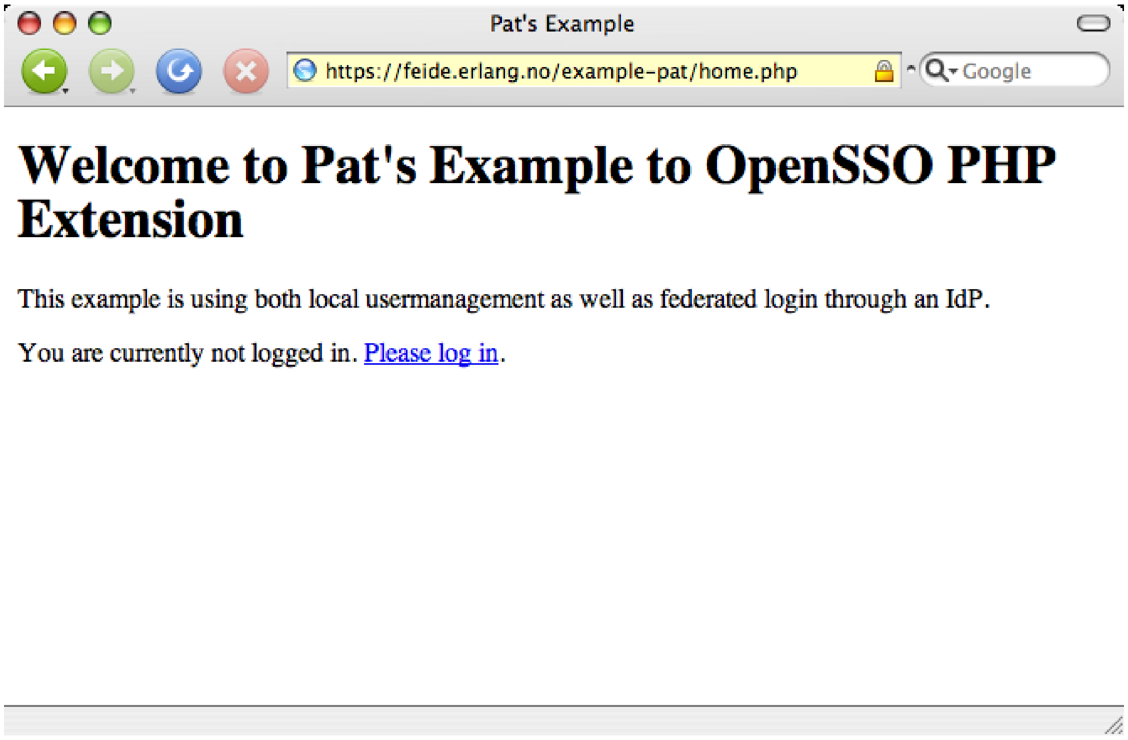
Start with editing the config file (see the section called "Configuration") to use these two plugins: `namemapping/database` and `sessionhandling/phpsession`. Remember to setup the database as described earlier to be able to use the `namemapping/database` plugin.

This example contains a set of files, all located in `example-pat/`. You do not need to edit any of the files, it will adapt to your installation based on the config file.

Warning

If you have tested another example earlier and not properly logged out, you need to clear your cookies before testing this example.

Start by accessing the URL pointing to the `home.php` file of the example, similar to `https://feide.erlang.no/example-pat/home.php`, but at your server.

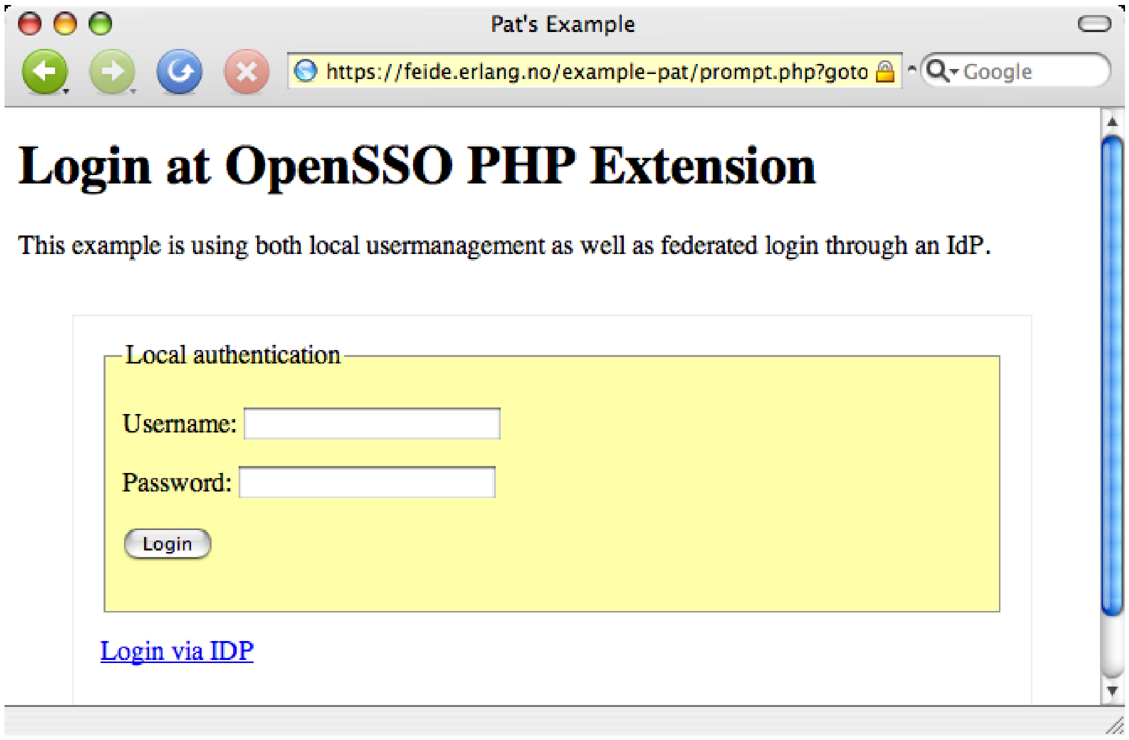


Welcome to Pat's Example to OpenSSO PHP Extension

This example is using both local usermanagement as well as federated login through an IdP.

You are currently not logged in. Please log in.

You will see a page like the one above, saying that you are not authenticated. Click on the "Please log in" link.



Login at OpenSSO PHP Extension

This example is using both local usermanagement as well as federated login through an IdP.

Local authentication

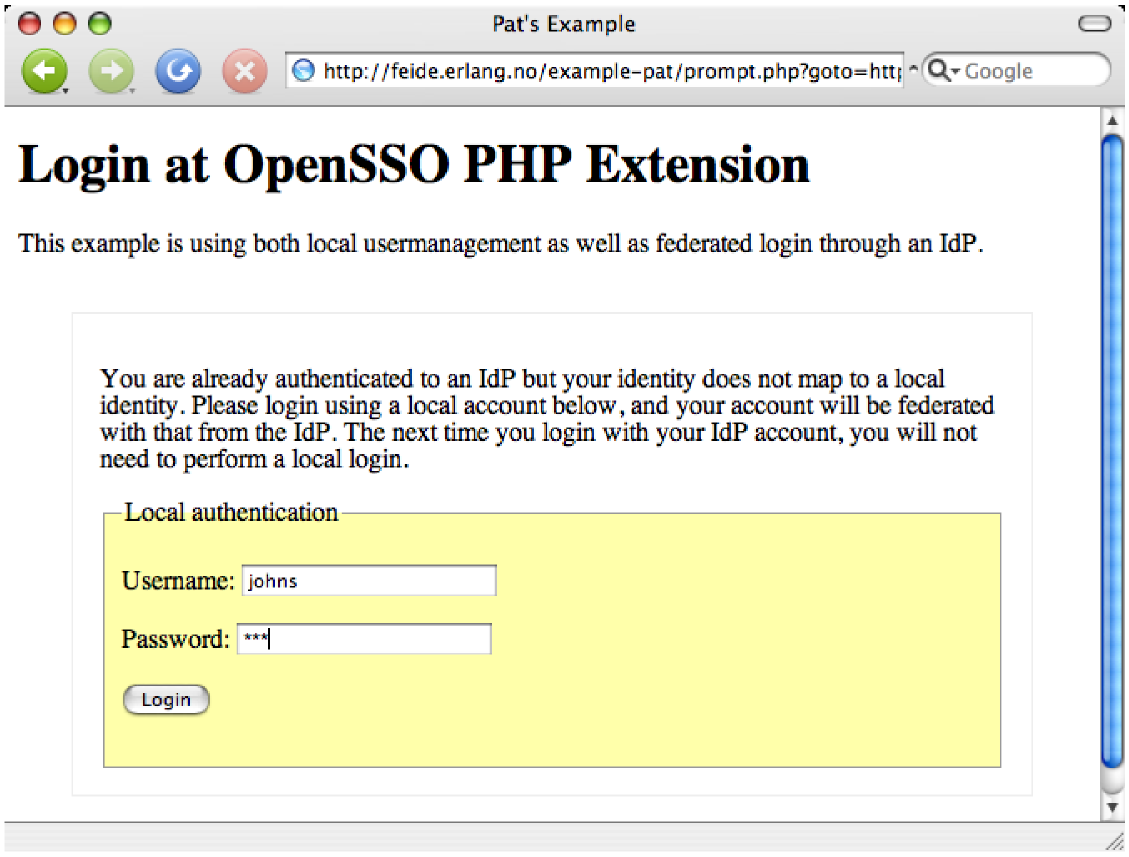
Username:

Password:

Login

Login via IDP

What you see is a local login page (`prompt.php`). All the users have a local username and password, and you may log in locally without communicating with an IdP. But, off course we want exactly that, communicating with an IdP, so we click on the "Login via IdP" link. If your meta data is configured correctly you will now see the login page at your IdP, please enter your credentials, and you will be redirected back to the example.



Login at OpenSSO PHP Extension

This example is using both local usermanagement as well as federated login through an IdP.

You are already authenticated to an IdP but your identity does not map to a local identity. Please login using a local account below, and your account will be federated with that from the IdP. The next time you login with your IdP account, you will not need to perform a local login.

Local authentication

Username: johns

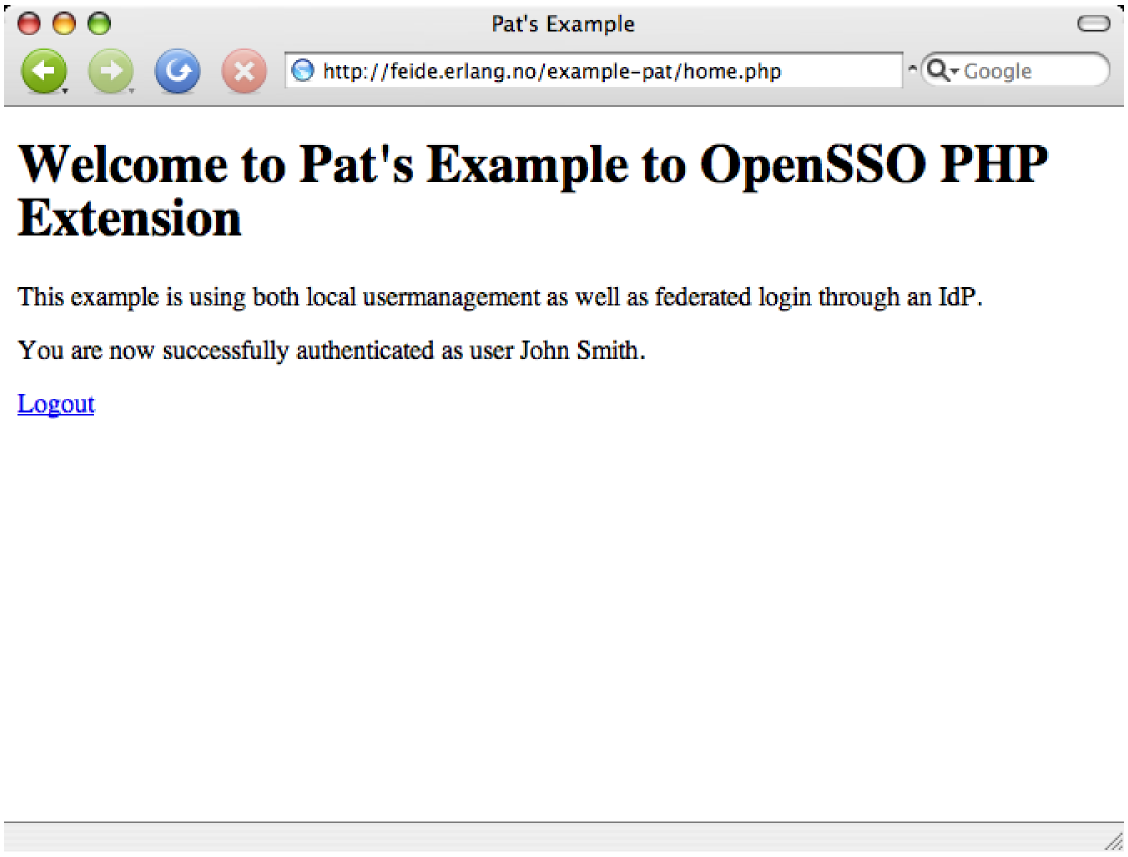
Password: ***

Login

You are now back at the example service, but Hey! - Why are you presented with a login sreen? You have already logged in at your IdP! The reason is simple, even though the IdP have presented your identity, the service needs to map that identity to a local user account. If you log in a local user account, a namemapping entry will be stored in the database, mapping the identity from your IdP to a local identity.

Tip

This example demonstration manually federating users (creating mapping between IdP and SP identities), but there also are other ways of doing federation. You may perform auto-federation, mapping to a local user account based on a provided attribute. You can also do auto-account-creation, automatily creating a local account the first time the user logs in, and then federate it automaticly (then there is no need for local passwords). You can also do batch federation, having a script updating the namemapping table regularly. Batch federation requires you to have access to both IdP and SP data storages, not always an option.



Welcome to Pat's Example to OpenSSO PHP Extension

This example is using both local usermanagement as well as federated login through an IdP.

You are now successfully authenticated as user John Smith.

Logout

As you see, you are successfully authenticated. Now, logout, and do the thing above one more time! What happens? As your user accounts already are federated you will now be directly authenticated from the IdP, not asked to log in locally. Cool, isn't it?