

Authenticating from multiple authentication sources in a collaborative work platform: the Picolibre & Shibboleth case study

Authors

Quang Vu DANG (IFI)

Olivier BERGER (GET/INT)

Christian BAC (GET/INT)

Benoît HAMET (phpGroupware)

Abstract

This paper presents a proposal to address the need for multiple authentication sources for users of collaborative work platforms based on phpGroupware, such as Picolibre.

The proposed approach, developed for the needs of GET and Picolibre, relies in a generic solution for integration of phpGroupware servers in a Shibboleth infrastructure. We have developed new phpGroupware adapters for this integration, which we hope to contribute to the phpGroupware project.

This document should serve as a basis for discussion among interested phpGroupware developers and users, in order to validate the level of generality of the proposed approach, and eventually decide of the adoption of the proposed modifications for integration in the phpGroupware standard code-base.

We hope that this approach can also help maintainers of other collaboration platforms, who want to integrate a park of deployed platforms with external user identification and authentication services, get a better view of solutions available with Shibboleth.

1. Introduction

The Groupe des Écoles des Télécommunications (GET) is composed of several engineering and business schools together with research centres in Paris (ENST), Brest (ENST Bretagne) and Évry (INT), in France. The research teams are made up of more than 600 full-time research equivalents.

The range of the researchers' expertise, from technologies to social sciences, enables the integrated approach so characteristic of GET research and fosters its adaptability to new application sectors and new usages in response to current challenges in the field of Information and Communication.

Starting in order for use as a pedagogical platform, Picolibre [1] is a libre-software system developed at GET, and released under the GNU GPL license. It provides a Web-based collaborative work platform built on top of phpGroupware¹ and other libre software tools. Picolibre provides project hosting facilities for small teams of software developers, which was mainly oriented to teaching and research environments².

Picolibre integrates several libre software Web applications, but lacked some features in the current stable version, like a wiki engine. However, we have developed an in-house GET platform, based on Picolibre components, integrating new services like a Wiki engine, or a Webdav server, called ProGET [2]. We seek to integrate these new features in a new generation Picolibre, making it a more generic and complete platform, and available for all as libre software.

Several Picolibre platforms have been deployed at GET, and developers or researchers may be using services of several such platforms, while working on projects initiated on these different platforms.

At the present time, different accounts may be created on these platforms for the same person. Here comes the need of Single-Sign-On (SSO) facilities between these platforms.

GET is in the process of deploying soon a federation of authentication systems and applications, based on Shibboleth, for a better integration of its Information System, which is at the present time distributed among the different schools.

Shibboleth is an infrastructure designed to build an identity federation allowing applications and identity providers to share and exchange attributes concerning user profiles, in order to facilitate user identification and authentication in the realm of a deployed identity federation (SSO, etc.).

We want to investigate the use of Shibboleth also by PicoLibre, but we should note that users of the Picolibre platforms may be :

- either GET agents or students, who are already registered in GET's "company directories" (so, who will be known to Shibboleth),
- or external contributors, outside GET (unknown to the GET Information System).

Nowadays, PicoLibre is not yet interfaced with GET's company directories to authenticate its GET users, and manages its own directory. Here comes the need to develop adapters to integrate Picolibre platforms in the coming Shibboleth federation.

But even if GET users are recognised by Picolibre, through its use of Shibboleth, we will still have to support other users not known of GET's directories. Such requirement will also be described in our proposed solution.

1 <http://www.phpgroupware.org/>

2 The reader will find more details about PicoLibre at <http://www.picolibre.org/>

The same issues, which are addressed here for Picolibre users and GET, may be found also for other networks of collaborative work platforms, for instance among libre software development communities, for authentication into the various software development platforms they use (Gforge, Trac, etc.).

2. Shibboleth and SSO Service

Shibboleth³ is a complete open source platform developed for project "Internet2"⁴, aiming at building the federation of identity for education institutions and their partners. It is based on the SAML standard (Security Assertion Markup Language), which defines the assertion of authentication and exchange of attributes of users. It supports the SSO service and the authorization, allowing the definition of access control privileges to Web resources, by using user-associated attributes.

The basic architecture of Shibboleth has three components:

- Identity Providers (IdP),
- Service Providers (SP),
- and a "Where Are You From" service (WAYF).

Identify Providers (IdP) are responsible of user authentication by using an existing SSO service, for example Central Authentication Service (CAS)⁵ and to provide the user attributes for the access control process. They manage a directory of users, attributes, and eventual passwords.

Service Provider (SP) is the part of Shibboleth (written in C/C++) managing access to the resources which are requested by the users in the federation. The Apache web server "plugin" *mod_shib* is a module which allows Web pages access control, basing its decision on using the user attributes values defined in the IdP.

The service "**Where Are You From**" (WAYF) is an additional component in the federation, which helps the user to explicitly choose his/her IdP "of origin", selecting the place where he/she may be known as a source of authentication.

When a user wants to use a service offered by the members of the federation, the service's Shibboleth SP component redirects the user towards his/her IdP, or a WAYF service for choosing his/her preferred IdP. Then he/she will authenticate to the SSO service which is indicated by this IdP (for instance a CAS server where a password will be prompted).

After successful authentication, he/she will be redirected automatically back to the requested service, but being authenticated, this time.

After having authenticated the user, the SP then requires some informations describing the user, and filters these informations for the authorization process. This information is then sent to the service's Web applications in the form of HTTP headers (through the *mod_shib* adapter, for instance, in Apache).

³ <http://shibboleth.internet2.edu/>

⁴ <http://www.internet2.edu/>

⁵ <http://www.ja-sig.org/products/cas/index.html>

The whole system is described in the following figure (1), describing the whole Shibboleth protocols :

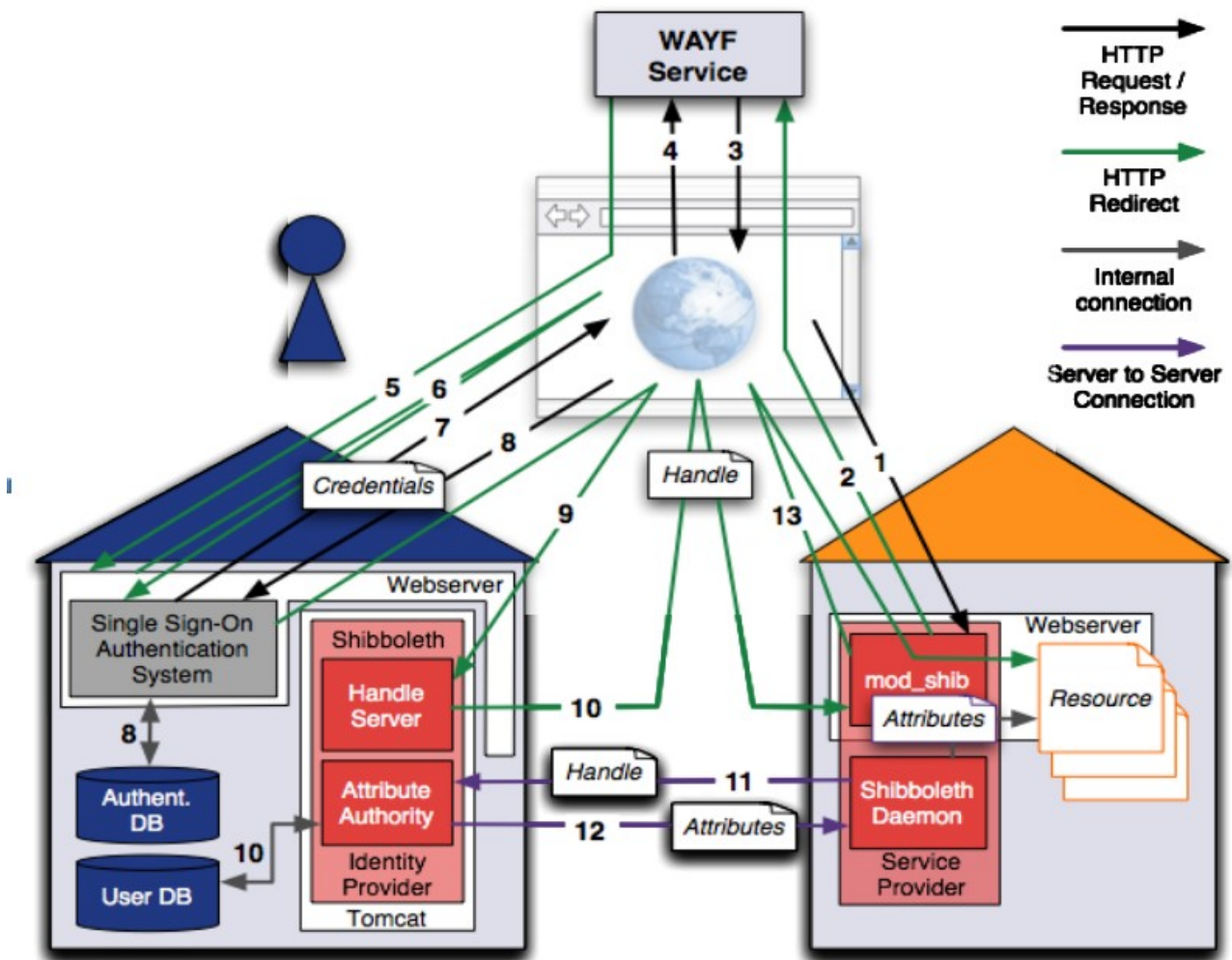


Illustration 1: Shibboleth protocols

This figure originally available at <http://www.switch.ch/aai/demo/Demo3.png> :
 © SWITCH (The Swiss Education & Research Network)

Shibboleth offers several advantages like the possibility to distribute a user-base among several directories (IdPs), that applications themselves don't keep local authentication tokens like passwords, or the fact that each applications may identify a user based on a different set of predicates on its Shibboleth attributes. This gives the whole system a lot of flexibility, while relying on a secure distributed and trusted architecture, with demanding user account management procedures on the IdPs. Discussion of all this is out of the scope of the present article.

3. Enhanced authentication in PicoLibre

At the present time, PicoLibre users can authenticate against the phpGroupware application (and its underlying database or LDAP directory), which provides the basis for users management in PicoLibre. They login to phpGroupware when they want to access the “virtual desktop” homepage of PicoLibre, which contains the list of the projects to which they collaborate.

But they may also authenticate directly to one of the other components integrated in the platform, residing on a same Web server, such as Sympa⁶ (the mailing list manager) or others to come in the future (like Twiki⁷), without going first to the phpGroupware login page.

3.1. Standard authentication Scheme in phpGroupware

In general, phpGroupware handles access-permissions to its applications in an autonomous way, for locally recognised users, basing itself on a local "account", stored in a "directory" operated for instance by a RDBMS like MySQL, or by a LDAP directory.

To be allowed to access phpGroupware, the user should have an account which determines the user's profile: the access rights for a list of the modules that the user may use.

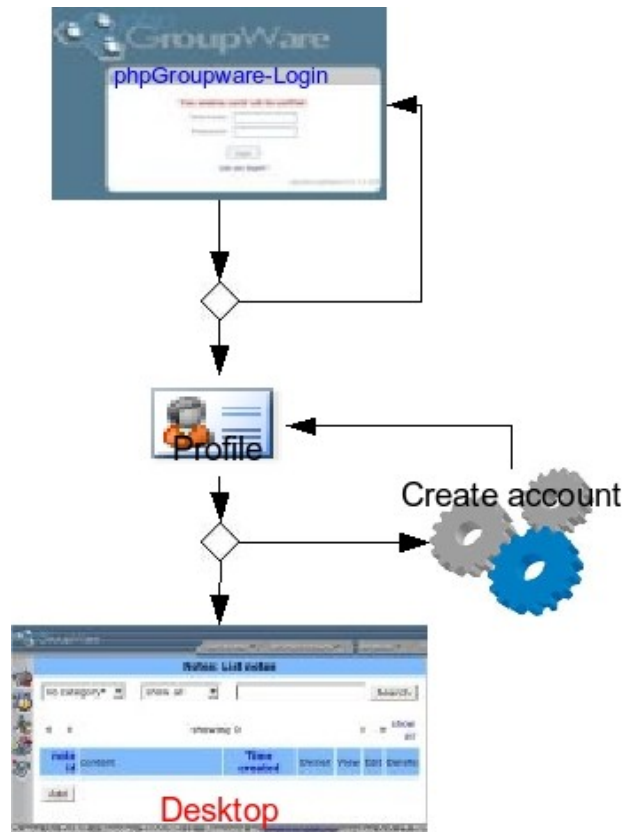
In phpGroupware, access to applications is a three phases process:

1. Authentication: verifying that the user is actually the owner of a user account, by using an instance of the *auth_* class which will retrieve the account, and validate the password against the local authentication directory.
2. Identification of the user's profile: this phase will be handled by an instance of the *account* class after it has been authenticated successfully. Here, if phpGroupware is setup in "auto-create" mode and no profile is already existing in the local directory, a new profile can be created automatically, by default ("guest" access for instance).
3. Creation of a work session: basing on the user's profile which has been retrieved, phpGroupware will create a work session, and grant access to the phpGroupware modules that the user is allowed to use.

Here, the phpGroupware authentication phase only uses a "local directory", as configured by the administrator during the phpGroupware server configuration phase.

6 <http://www.sympa.org/>

7 <http://www.twiki.org/>



Depending on the physical implementation of the local accounts directory used by phpgroupware (MySQL, LDAP, ...), it is possible to share the user's profile with other applications deployed in the same networked environment as the phpGroupware server, providing that necessary administrative policies are adopted, and that custom technical adapters are developed, or configuration decisions are taken. But this is quite demanding with respect to the administrators' skills. Using the same LDAP directory as a user account directory for several high-level Web applications may be such an easy solution (actually used at the moment in PicoLibre), but may not scale and adapt to all security concerns, anyway.

So there's, at the moment, no “elegant” SSO service facility in phpGroupware, which would allow phpGroupware to grant, to users already known in other parts of the organisation's information system, a "transparent" access to its applications (unless they share the same “back-end” LDAP server for instance, as explained above).

3.2. Shibboleth+Apache as an integrator and SSO service

An other way to assemble all the current authentication mechanisms in the various Web applications installed on the same PicoLibre system, is the use of a common “front-end”, i.e their common Web server's authentication system (here we consider only Apache, as a standard reference tool).

But if (in the future) the applications composing a single PicoLibre platform are actually deployed on several Web servers, we need a more advanced mechanism for sharing this authentication.

We still need also, as described above, SSO with other applications deployed throughout GET, outside the Picolibre platform, to which users will have already authenticated.

Shibboleth can come and help solve all these needs. And hopefully many Web applications we use, like Sympa, or Twiki are now becoming compatible with Shibboleth. Thus, Apache and Shibboleth will be able to act as the integrator of their authentications mechanisms (using a distributed Web service approach).

But unfortunately, phpGroupware's authentication mechanisms come short in such a situation. We need to develop a new phpGroupware identification and authentication adapter for the Apache + Shibboleth combination.

3.3. Mixed environment, and legacy

Picolibre may be using the GET Shibboleth federation once adaptors have been added to all its applications. But, as we have described our users typology above, Shibboleth can't be the exclusive authentication mechanism used. We are not in a typical "intranet" or "extranet" system, and have both "company" users, and "external" users. We will then need some way to "bypass" Shibboleth for some of our users.

Also, one issue which has to be solved when Shibboleth (or any such external authentication mechanism) is used, is the local *mapping*, in the applications, between the way users are recognised in Shibboleth, identified, and the internal reference of the local profile (or account) in the application.

Of course, if Shibboleth is deployed prior to setting-up the Picolibre platform, and all its users will be known in Shibboleth, and only Shibboleth users are recognized by the applications, then such mapping is trivial. But it gets much worse if Shibboleth is deployed on an existing environment with many legacy accounts already existing in Picolibre.

Also, Picolibre users working or studying at GET may be known in various Shibboleth IdPs inside GET. But it is not obvious yet, at the present time, that they may be recognised by standard set of attributes fitting the needs of Picolibre.

Having considered all the constraints above, we propose to integrate Picolibre (hence phpGroupware) with Shibboleth with a flexible approach. In particular, here, we try to facilitate the progressive integration of existing deployed phpGroupware instances, thus diminishing the migration burden for administrators and users (accounts re-creation, etc.).

As a result, the design of the new Picolibre authentications system (including the phpGroupware Shibboleth adapter) will then need to support these cases :

- new users, who are known in GET's company directories (like GET agents and students registered in one of the Shibboleth IdPs), but not yet having a developer account in Picolibre, who will be able to create a new account in Picolibre/phpGroupware,
- “legacy” users of Picolibre with an account in phpGroupware, who will still access Picolibre, and :
 - if they also have an account in Shibboleth at GET, be able to *map* their legacy Picolibre account to the new Shibboleth identity (for doing SSO with the CAS services of the IdPs),
 - if not (for people external to GET), still be able to use Picolibre, as before, “bypassing” the Shibboleth SSO engine,
- new users external to GET, who will still be able to apply for registration in phpGroupware, as before.

The situation should be similar for any other authentication mechanism than Shibboleth, to which phpGroupware would authenticate. We then tried to propose a standard *mapping* mechanism which would not be too specific to Shibboleth.

4. Implementation of the phpGroupware Shibboleth adapter

We will describe in this section the way we have implemented the Apache + Shibboleth adapter in phpGroupware, taking into account all the constraints described above.

4.1. Using Apache-based authentication in phpGroupware

By using an Apache authentication method, phpGroupware will not authenticate users internally, in its accounts directory (LDAP, MySQL, ...). Instead of that, it will depend on the Apache session's environment variable *REMOTE_USER* (which will hold something like a user's logname, or email,...), which is defined when the HTTP transaction is authenticated by the web server⁸.

The benefit of this approach is the availability of many authentication schemes, using the existing Apache modules such as *mod_auth_ldap*, *mod_auth_mysql*, *mod_cas*, *mod_shibb*, etc.

PhpGroupware will then take full advantage of this versatile integration mechanism with external authentication sources like Shibboleth.

However, depending on the browser's features, and specific mechanism used, in certain cases, the identity of the user may be “hidden” in the web browser's internal state : he/she may have initiated a session, but may not be able to logout from phpGroupware unless the browser is closed⁹.

⁸ With the “basic” authentication mechanism, when no advanced Apache auth. driver is used (like *mod_shib*), this will result in the prompt of a login + password dialog-box on the Web browser.

⁹ Some current Web browsers versions don't support easy HTTP session's *REMOTE_USER*

For instance, the configuration of the Apache web-server for a phpGroupware instance may be such as (for auth. against a LDAP server) :

```
<location /phpgroupware>
  AuthType Basic
  AuthName "phpGroupware"
  AuthLDAPEnabled on
  AuthLDAPAuthoritative on
  AuthLDAPURL ldap://my.openldap-server.com/dc=my_org,dc=org?uid
  require valid-user
</Location>
```

4.2. Mapping Apache session's REMOTE_USER to a phpGroupware account

In our case, with Shibboleth, we want phpGroupware to take part in a federation of identity providers, which consists in several sources of identity attributes.

Once a user has authenticated to Apache (through mod_shib), we need to determine its profile, by retrieving its phpGroupware account, based on the use its attributes in Shibboleth. The choice of the attributes to use is difficult.

For instance, two different users may appear as having the same "identity" in one specific attribute's values on different IdP (same *uid* number in different LDAP servers, for instance). Then, if we used the existing "trivial" mapping of Apache REMOTE_USER values to phpGroupware accounts IDs, it wouldn't work. We then need to use another attribute, say the *email*, which is supposed to be of unique value among all the IdPs of the federation (or some ID card number, etc.).

But let us now suppose that the chosen attribute is the email; then, depending of the identity policy enforced in the federation¹⁰, a single person may belong to two different sources of identities where he/she is known with two different emails: for instance, a departmental email (with a sub-domain in the rightmost part), and a company-wide email (with only the master domain name of the company).

To better support these situations, we felt the need to develop a new (optional) *mapping* mechanism, active during the identification phase in phpGroupware, which would use a *mapping table* for projection of these different attribute values to a single account ID in phpGroupware.

There may then be two modes of operation available in phpGroupware, configured by the administrator :

- the old "trivial" mapping, for sites where REMOTE_USER is a unique ID in the federation of identities (using current phpGroupware implementation).

re-initialisation by default

¹⁰ It is not clear yet to the authors if such situations may arise when the full Shibboleth infrastructure will be rolled-out through the GET, as we are not directly associated to the deployment project. Anyway, even though this case may not be frequent with Shibboleth, it will probably happen with other Apache based auth. sources in large environments, or during migration phases, for instance.

- the new *mapping* mechanism, for sites where each user does not necessarily have a unique ID which can be identified in the company's Information System.

4.3. Additions to phpGroupware

PhpGroupware's code will be enhanced with a new class *mapping* that we have developed (which is described in more details in the Annex).

After having passed the phases of authentication (with Apache and Shibboleth, for instance) a user will be directed to the new *login.php* equivalent script, in the `/phpgroupware/phpgwapi/inc/sso` directory, which will realise the mapping, and the creation of the phpGroupware work session.

If the search for an existing suitable mapping is not successful :

- a new account can be created,
- or a new mapping can be created to an existing account in phpGroupware (verifying, on creation, that the user owns the password to this existing account).

If phpGroupware allows users to create the accounts (as defined in phpGroupware's configuration : "Autocreation of account"), the script `create_account.php` provides an interface for creating the new account, based on informations provided by the external auth system (in the case of shibboleth). It will help retrieving various user's description fields like names, email, IDs, etc.

If phpGroupware is configured to support the new "mapping by table" feature, the script `create_mapping.php` provides the function for creating a new mapping if the user already had an existing account in phpGroupware. During this process, the user will have to authenticate to the legacy account in phpGroupware, before creating this new mapping, in order to ensure that nobody can hijack anyone's existing account when a deployed Picolibre platform will enter the new Shibboleth federation.

If phpGroupware is configured to use the "trivial" mapping mechanism, no such authentication to the legacy account will be done, as we fully trust the IdPs, and consider that the matching of Shibboleth attributes to the local accounts is non-ambiguous.

In the case there would be a successful match for trivial mapping for most users, but only a small number of failing cases, then a "sequential" mapping mechanism could be activated, providing that the administrator has carefully checked for absence of potential "hijacking" risks. In such cases, both mechanisms would then be applied sequentially : trivial mapping first, then mapping "by table" if no success.

Several important configuration options of phpGroupware used here may be such as the following in `setup/templates/default/config.tpl`:

```
auth_type = "remote user"
```

```
mapping_type: choice between
    "All",
    "Unique ID",
    "Mapping Table"
mapping_field: choice between
    "uid",
    "email",
    "account_lid" (default)
...
```

We want to stress that this new phpGroupware user account identification, through an optional mapping phase, is not specific to the use of Shibboleth as an external authentication method. It may become useful also for other sources of authentication through Apache (or with other mechanisms), so we propose that it will be integrated in the standard code-base of phpGroupware's core-API module.

Thus phpGroupware will become much more inter-operable with the whole IS of a company, and less oriented to use as a specialised dedicated and isolated groupware server.

5. Configuration of phpGroupware's access protection in Apache

Access to phpGroupware's Web pages and to associated Web applications of PicoLibre can be configured through careful customisation of Apache configuration directives.

As we explained above, we may be using phpGroupware in an "intranet" environment, or a mixed environment such as the one for PicoLibre. In a mixed environment like PicoLibre's, where we will use authentication through Apache and Shibboleth, Shibboleth will only recognize company agents, and not external "contributors" to hosted projects. The same problems stand for every other authentication mechanisms than Shibboleth, actually, when in mixed environments.

Through specific configuration directives, we can devise different access points for the same Web application, and choose to make access control by Apache mandatory (called "Full Apache"), or optional (resp. "Semi Apache")

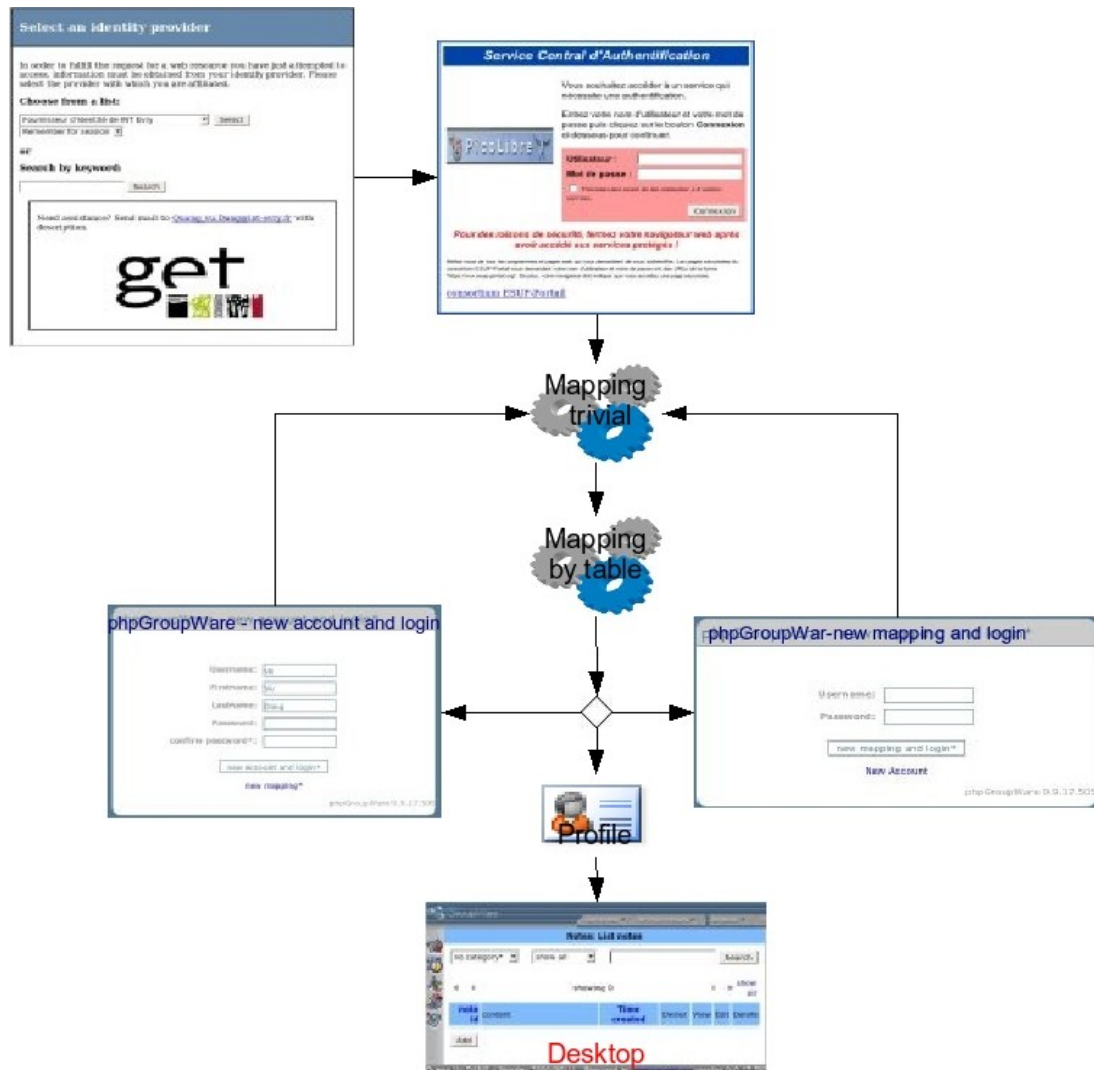
5.1. "Full Apache" access control

In this case, Apache acts as a "wrapper" around the *whole* phpGroupware application, protecting all its accesses. Every access to phpGroupware must go through Apache authentication module (hence Shibboleth). Only users already known to the identity federation will be granted access.

This can be achieved through an Apache server configuration like the following :

```
<Location /phpgroupware>
    ...[any auth. mechanism here]
    require valid-user
</Location>
```

When users are authenticated, they will proceed to the optional mapping phase, which will eventually help retrieve any existing legacy phpGroupware local accounts (see figure below for the resulting whole process in “Full Apache” mode).



5.2. Semi Apache

On the other hand, phpGroupware may be accessible also without first authenticating to Apache (“bypassing” Shibboleth). This may be useful to be able to setup, at the same time :

- an authentication on the local phpGroupware accounts directory, with a traditional phpGroupware auth method (for external users outside the company, contributors, ...)
- and an authentication against Apache (and Shibboleth) as well (for instance for all regular "Intranet/extranet" access by company agents, students), with SSO option if available.

In Apache configuration directives, for instance, the “top-most” location /phpgroupware/ itself would be “world accessible” (in the limits of phpGroupware's own login script, and protections in the PHP code).

Then the configuration of the specific part providing authentication through Apache's auth modules should be available under a subdirectory, for instance /phpgroupware/phpgwapi/inc/sso:

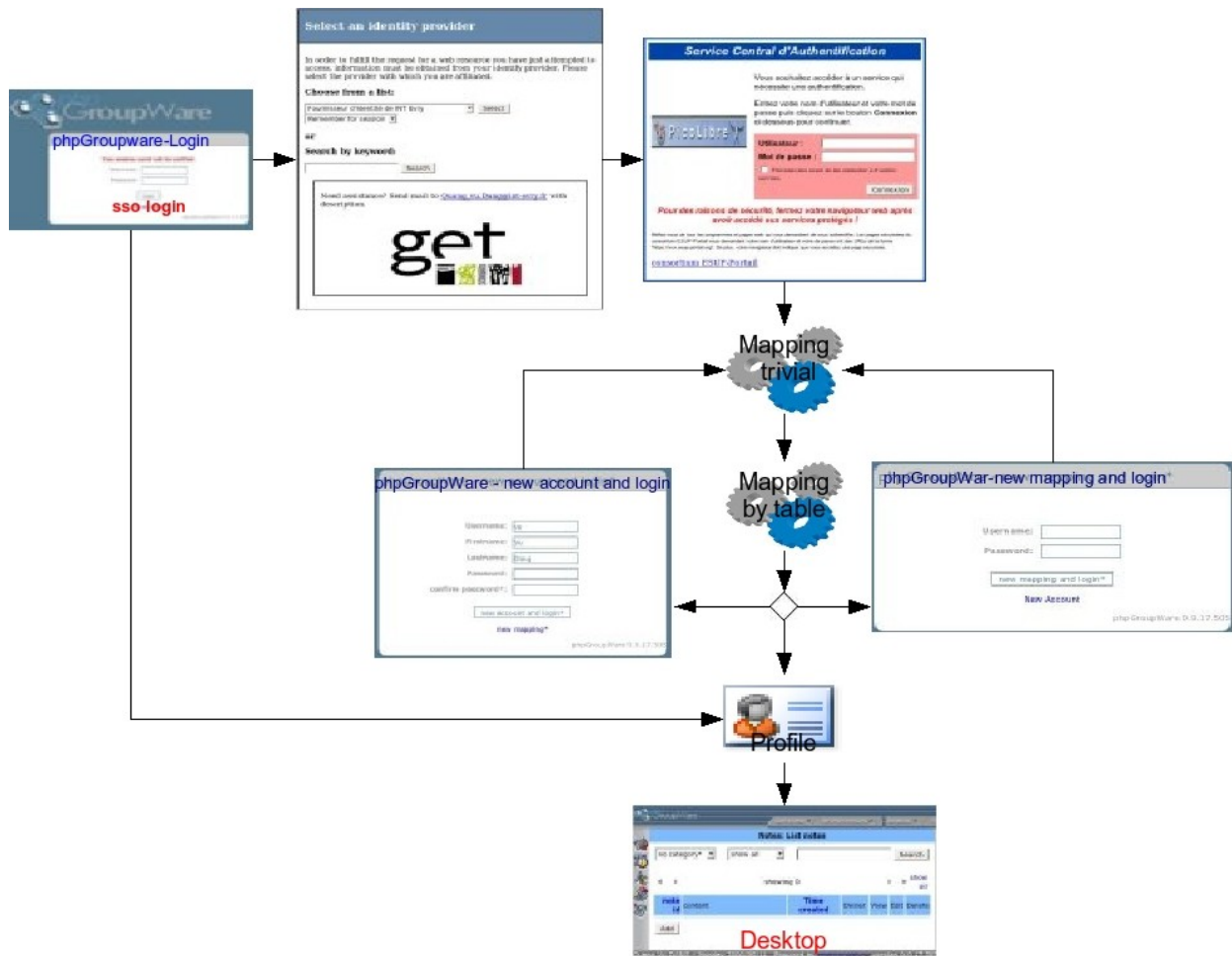
```
<Location /phpgroupware/phpgwapi/inc/sso>
  ...[any auth. mechanism here]
  require valid-user
</Location>
```

So, to access the phpGroupware server, the user would be free to choose whichever method suits him/her most, by going to different entry points (different URLs), for instance :

- <http://server/phpgroupware/login.php>: which would propose the standard phpGroupware login + password PHP dialog. Used by local users, and which would also provide an alternate link (named "SSO", for instance) pointing to the other URL bellow,
- and http://server/phpgroupware/phpgwapi/inc/sso/login_server.php¹¹ for instance : which would be accessed from the SSO link above, or directly, whose sole purpose would be to direct to the Shibboleth infrastructure login and SSO pages (plus mapping).

The login process would then be the following :

¹¹ Note that the URLs provided here may be aliased through rewrite-rules, etc. for simplicity of use. The naming hierarchy used here reflects the physical distribution of the scripts in the phpGroupware installation.



6. Final phpGroupware setup with Shibboleth for PicoLibre

With Shibboleth acting in the Apache access control wrapper's for Web applications of PicoLibre, as described above, and with addition of the necessary mapping phase, phpGroupware can now use the SSO services available in the GET Shibboleth infrastructure. PicoLibre platforms will then participate to the Information System in a standard way for most of its users, accessing the identity federation to recognise them.

For the specific needs of PicoLibre platforms at GET, phpGroupware access will be configured to use the "Semi-Apache" authentication method described above, since we have users outside the boundaries of GET.

In the Apache configuration we will use the Shibboleth authentication :

```

<Location /phpgroupware/phpgwapi/inc/sso>
  AuthType shibboleth
  ShibRequireSession On
  require valid-user
</Location>

```

As we have explained above, the user known to Shibboleth may be known from several of identity sources (IdP), and we need to define which attribute will be used to identify the user, and define the mapping strategy to phpgroupware accounts in consequence.

We chose the email, so the AAP.xml configuration file of the Shibboleth SP associated to phpGroupware will be defined so as to transmit the value of the user's email for REMOTE_USER :

```
<AttributeRule Name="urn:mace:dir:attribute-def:mail" Header="REMOTE_USER">
  <AnySite>
    <AnyValue />
  </AnySite>
</AttributeRule>
```

The same kind of configuration can then be applied for access to other integrated applications of the Picolibre platform such as Sympa (or TWiki in the future) for them to benefit also from phpGroupware's or external applications authentication for SSO.

7. Conclusion

We have described a proposed method for integrating phpGroupware with Shibboleth to allow the use of SSO mechanisms in accessing phpGroupware, while supporting several authentication sources (various Shibboleth IdPs, and local accounts too).

The integration relies a lot on the use of Apache's authentication modules instead of proceeding with an internal phpGroupware auth. mechanism. There are very few specifics of Shibboleth in this respect, most of the issues being the same if phpGroupware uses other types of Apache auth. mechanisms.

We introduce several options for configuration and adaptation to other environments, in order to achieve the most generic solution. They help adapt to several kinds of situations, like the availability of a fully operational Shibboleth environment, or the support of an existing deployed phpGroupware user base, potentially not already in sync with the Shibboleth deployment.

Integration of the new modules developed in phpGroupware has been implemented with the current 0.9.16.010 version of phpGroupware¹², and avoiding modifications of its architecture.

We have successfully tested on a prototype system on two phpGroupware and Picolibre platforms, with the Shibboleth infrastructure of GET/INT.

Further tests will be needed, but we are satisfied with the proof of concept obtained, which offers us a migration path to an easy integration of deployed Picolibre platforms in the GET IS, while keeping existing legacy accounts.

The SSO facility obtained will help design new networks of collaborative platforms that will hopefully offer greater usability and flexibility, for wider adoption both inside companies or among creative communities on the Internet.

¹² and may be partially integrated also in version 0.9.18 at the time you read this document.

Bibliography

[1] Cousin E., G. Ouvradou, P. Pucci and S. Tardieu, 2002, *PicoLibre a free collaborative platform to improve students skills in software engineering*, in: 2002 IEEE International Conference on Systems, Man and Cybernetics, Vol.1, IEEE, p. 564-568.

[2] Berger O., C. Bac, and B. Hamet, 2006, Integration of Libre Software Applications to Create a Collaborative Work Platform for Researchers at GET, International Journal of Information Technology and Web Engineering 1 (3), 2006.

Annex

Components added

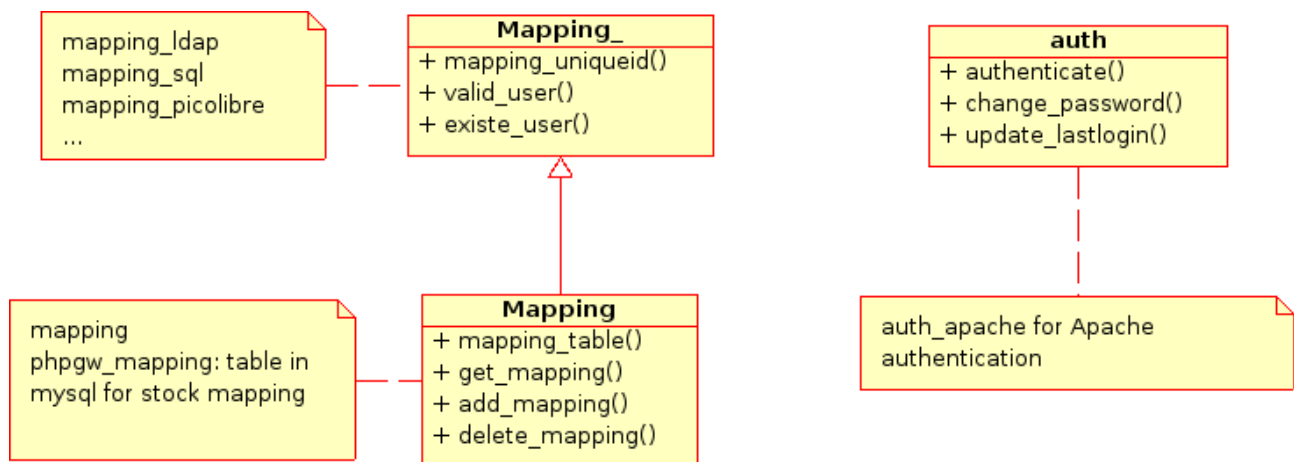
In package phpgwapi

In this package we add the *auth* class in the file *auth_remoteuser.inc.php* which corresponds to the method of authentication through Apache. This class does not do a lot, it is only overloading the base class with the trivial process of authentication in phpGroupware.

For the mapping phase during identification of the user, we add the class *mapping* in the file *mapping.inc.php*. It inherits from the class *mapping_* which is based on the existing account in phpGroupware, to make the mapping "trivial" and to validate the account.

Class list :

```
class auth : auth_apache.inc.php
class mapping_ : mapping_ldap.inc.php
class mapping_ : mapping_sql.inc.php
class mapping_ : mapping_picolibre.inc.php
class mapping : mapping.inc.php
```



New "apache" phpgrouware module

We add a new Apache package for the authentication though Apache.

Scripts :

- */phpgwapi/inc/sso/login_server.php* : for login process,
- */phpgwapi/inc/sso/create_account.php* : interface allowing the user to create an account
- */phpgwapi/inc/sso/create_mapping.php*: interface allowing the user to create a new mapping
- */preferences/mapping.php*: manage mappings (Allow, Deny, Delete a mapping)

New database table

We add the table *phpgw_mapping* (attributes : *user_ext*, *location*, *auth_type*, *status*, *account_lid*) to make mapping by table. It links values of *REMOTE_USER* to user local phpGroupware accounts.

Components modified

Setup / configuration

We modify the configuration of phpGroupware in module */setup/templates/default/config.tpl* :

- Add an option for the type of authentication though Apache : *auth_type = remoteuser* (only choice available at the moment, but present for future needs)
- Add an option Allow fallback authentication method (at the time of writing only remote user is allowed)
- Add a parameter *mapping_field* indicating the account if to make mapping "trivial"
- Add options *mapping_type (all, id, table)* indicating the type of mapping

phpGroupware modules

- */login.php* : modified to make mapping if *auth_type = remoteuser*
- */admin/inc/hook_deleteaccount.inc.php* : remove the corresponding mappings

Configuration decisions

If using Apache server's auth modules (*mod_auth_ldap*, *mod_auth_mysql* ...), declare the module configuration to protect phpGroupware.

For exemple with *mod_auth_ldap*

```
<location /phpgroupware>
  AuthType Basic
  AuthName "phpGroupware"
  AuthLDAPEnabled on
  AuthLDAPAuthoritative on
  AuthLDAPURL ldap://my.openldap-server.com/dc=my_org,dc=org?uid
  require valid-user
</Location>
```

Corresponding configuration in phpGroupware

```
auth_type = apache (for Full Apache)
auth_type = sql,ldap... (with option semi_apache = yes : Semi apache)
mapping_type=id,table,all
mapping_field = account_lid (By default REMOTE_USER=uid)
```

When using Shibboleth(mod_shib)

Install SP and indicate the attribute chosen for REMOTE_USER (uid, email...)

For example to put the value of the email for REMOTE_USER.

```
<AttributeRule Name="urn:mace:dir:attribute-def:mail" Header="REMOTE_USER">
  <AnySite>
  <AnyValue/>
</AnySite>
</AttributeRule>
```

Declare mod_shib with Apache server to protect phpGroupware

```
<Location /phpgroupware>
AuthType shibboleth
ShibRequireSession On
require valid-user
</Location>
```

Of course the Shibboleth SP will need to be registered within the existing federation.

Configuration in phpGroupware :

```
auth_type = apache : Full Shibboleth
auth_type = sql,ldap... with option semi_apache = yes : Semi Shibboleth
mapping_type = id , table , all
mapping_field = account_lid or the other (Depend with REMOTE_USER)
```