DrupalShibbolethReadmeDev

Drupal **shib_auth** module enables Shibboleth authentication for Drupal CMS.

Installation and bootstrapping

> **Warning**
>
> The following documentation assumes that
>
> • You understand how Shibboleth works
> • You have successfully installed and configured Shibboleth SP on your host running Drupal.

Installing the module

1. Download module source for your Drupal version from the project page.
2. Uncompress archive to the `modules/` directory
3. Enable module at **`Administer -> Site building -> Modules`**

Compatibility

The module assumes that you use Shibboleth SP 2.x. It's possible to use the module with Shibboleth SP 1.3, but that version is not supported anymore.

Upgrading the module

> **Figyelem**
>
> If you upgrade from 3.x (or previous versions) to 4.x, **you must update the database**. After overwriting the files in the `modules/` directory, open up

> `YOUR_DRUPAL_INSTALLATION/update.php` in your browser and run update for the `shib_auth` module. **This is required to let your user info persist.**

The upgrade script makes slight changes to the database. You only need to run this script once, however running it several times does no harm to your installation apart from showing some SQL errors.

> **Note**
>
> although the upgrade script was designed to be robust, please **back up your database** before upgrading.

Moving from Drupal 6 to Drupal 7

To migrate your working installation to D7, first you must update your module to the latest 4.x version. After that, you can perform the Drupal update.

Get it running

Configuring Shibboleth

You should be familiar with protecting resources with Shibboleth before using this module. (See Shibboleth Wiki for comprehensive information) Please check that Shibboleth authentication is working for the location of your Drupal installation and all the necessary attributes are exported to the headers. You can enable DEBUG mode to dump the whole **$_SERVER** array. If you can see Shibboleth attributes there, you're fine.

In Shibboleth there are two modes for protecting resources:

- **Lazy Sessions**: session is only initiated if an application redirects user to the SessionInitiator URL. In this module, it is done by clicking the "*Login with Shibboleth*" link. Anonymous access is possible as well as using other authentication methods. This is what you want to use for a CMS that contains *public information*.
  - Detailed description of lazy sessions in Hungarian.
- **"Strict" Sessions** (normal sessions): users can only access Drupal content if they have a valid Shibboleth session. In this case, no anonymous access can be granted (not even read-only) and you can not use any auxiliary authentication methods. Most of the advanced features don't work with strict sessions. You should not use this unless you want to protect all the information in the CMS from viewing.

> **Note**

> after enabling "strict" sessions, you won't be able to login with admin user. Read on further how to give your own user administrator rights.

Example Shibboleth configuration

> **Note**
>
> this example uses lazy sessions.

/etc/shibboleth/shibboleth2.xml snippet:

```
<RequestMapper type="Native">
  <RequestMap applicationId="default">
    <Host name="your.host.name">
      <Path name="location_of">
        <Path name="your_Drupal">
          <Path name="installation" authType="shibboleth" requireSession="false" />
        </Path>
      </Path>
    </Host>
  </RequestMap>
</RequestMapper>
```

Apache config file snippet (ie. `/etc/apache2/sites-enabled/your.host.name`, or you can even use `.htaccess` without the `<Location>` tags):

```
<Location /location_of/your_Drupal/installation>
  AuthType Shibboleth
  ShibRequireSession Off
  ShibUseHeaders On
  require shibboleth
</Location>
```

DEBUG mode

If you enable DEBUG mode on the module configuration interface, you can dump the **$_SERVER**, **$_SESSION** arrays and additionally the **$user** object, if the user is logged in. This mode shows you all the available attributes and helps you diagnosing possible Shibboleth attribute problems.

- Keep in mind that some users might have a specific attribute while others don't.

Debug path prefix

Leave it empty, if you want to display debug information on every page. Enter the string `user/` (mind the trailing slash!) for displaying DEBUG messages only on paths below `user/*`

Adding a prefix is useful, if you want to enable debugging on an online drupal installation without littering all of the pages with the debugging information. You can set it to a non-existent node as well, in this case, the information will be displayed over the built-in 404 page.

Setting Shibboleth parameters for the module

Handler settings

If you are using lazy sessions, you have to define the Shibboleth SessionInitiator to which the user should be directed when she clicks on "Login with Shibboleth". The SessionInitiator can be called on a special URL which depends on your Shibboleth configuration.

If your **/etc/shibboleth/shibboleth2.xml** is something like this:

```
<Sessions lifetime="28800" timeout="3600" checkAddress="false"
          handlerURL="/Shibboleth.sso" handlerSSL="false"
          exportLocation="http://localhost/Shibboleth.sso/GetAssertion"
          idpHistory="false" idpHistoryDays"7">
  <SessionInitiator type="Chaining" Location"/Login" isDefault="true" id="Intranet"
                    relayState="cookie" entityID"https://idp.example.org/shibboleth">
    <SessionInitiator type="SAML2" defaultACSIndex="1" template="bindingTemplate.html"/>
    <SessionInitiator type="Shib1" defaultACSIndex="5"/>
  </SessionInitiator>
  <!-- other things -->
  <LogoutInitiator type="Chaining" Location="/Logout" relayState="cookie">
    <LogoutInitiator type="SAML2" template="bindingTemplate.html"/>
    <LogoutInitiator type="Local"/>
  </LogoutInitiator>
  <!-- other things -->
</Sessions>
```

Then your

- **Login URL** is: `http(s?)://your_fully_qualified_domain_name/Shibboleth.sso/Login`
- **Logout URL** is: `http(s?)://your_fully_qualified_domain_name/Shibboleth.sso/Logout`

> **Note**

> Most common errors are (please check Shibboleth documentation):
>
> • using *http* when your Shibboleth SP is configured for *https* only (either by Apache settings or the *handlerSSL* parameter)
> • you want to use another SessionInitiator (for example with Discovery Service), such as `/DS`

Attribute settings

Specify here the **$_SERVER** headers to look up the user's username and e-mail address. Please check **DEBUG** mode to look for the available headers. If you can not find the desired attribute, then something is wrong with your IdP-SP attribute release flow.

Both fields can have the same value, if you wish.

Redirecting authenticated users to HTTPS

It's a common problem if Shibboleth SP is configured on HTTPS only (`handlerSSL="true"`), while the site also works on plain HTTP. By using *Force HTTPS on login* feature, you can redirect your users to HTTPS at login time. This way you can have your anonymous users view your site unencrypted (which saves some CPU cycles), but once they login, they get redirected to HTTPS (which is the only secure use of Shibboleth SP).

> **Note**
>
> If you don't see Shibboleth attributes in DEBUG although you have double checked that you have set `AuthType shibboleth`, `require shibboleth` in your `.htaccess` file, you most probably need to turn this feature on.

Understanding features

Automatic user creation

Drupal CMS requires all users to be in its internal SQL database. If the module detects that no user exists in the database with the received Shibboleth user identifier, it creates a new (Drupal) user.

Drupal needs 3 pieces of information to create a new user:

- username
- e-mail address
- password

The module by default uses the username and e-mail address taken from the `$_SERVER` headers, see [Attribute settings](#bkmrk-attribute settings) right above. On new user creation a random password is generated. This can be overwritten by the user unless you disallow it by the userprotect module.

> **Note**
>
> if the user overwrites the password, she can log in with her username and the new password.

Using custom values

You may want to let your users to define their own Drupal usernames and e-mail addresses other than what was received from the IdP. If either *User-defined usernames* or *User-defined e-mail addresses* option is set, new users are presented a form to enter data at the first logon.

> **Info**
>
> • The module ensures that neither values are in use by existing users.
> • If you enable and later disable the options, the two behave differently. On subsequent logons:
> • the user-defined username is **preserved**
> • e-mail address gets **rewritten** (or the user gets a fatal error if the IdP does not provide the data)
> • Even if you enable user-defined usernames, the **IdP must send a unique identifier** which must be in the server variable defined for username. This feature just enables you to use opaque identifiers such as `eduPersonTargetedId`
> • User-defined e-mail addresses are not verified, only well-formedness is checked

Working with federated identifiers

If you enable the option *User-defined usernames* in the module configuration, every new user is presented a form to specify a Drupal username. This way you can work with opaque (federated) identifiers such as `eduPersonTargetedId`, as long as it appears in the $_SERVER variable holding the username. The only limitation is that the federated identifier cannot be longer than 255 characters, however it can contain characters otherwise not allowed in Drupal account names (such as exclamation mark, etc).

Disallowing password change

If you don't want to let your users to change passwords and log in with it, you may want to disallow password change.

1. Install Drupal User Protect module
2. At Administer -> User management -> User Protect -> Protected roles tab check **password** for the *authenticated user* role.
3. At Administer -> Permissions -> userprotect module: uncheck **change own password** for *authenticated user*
4. Log in with a normal account, go to "My account" -> Edit. You shouldn't see the possibility for changing password; except for the case when the user has user administrator rights.

Administering Drupal with strict sessions

If you use strict sessions, you can not log in with a password. You need to grant your own user administrator rights to administer the CMS.

1. Enable Shibboleth protection
2. Login with your own user credentials, so that your Drupal user profile is created
3. Disable Shibboleth protection
4. Login as 'admin', grant your own user 'Administrator' rights.
5. Enable Shibboleth protection
6. Login with your own credentials, you should have 'Administrator' rights now.

Pre-creating users

Versions before 4.0 allowed pre-creating users without any tweaks; if the username matched, the user was logged in.

Since 4.0 the module only logs in users who exist in `{shib_authmap}` and the update script only takes care of users tagged with 'shib_auth' in `{authmap}`. When there is no mapping in the `{shib_authmap}`, a new user is attempted to be created, which fails because of the mail being duplicated. So what was accidentally working with pre-created users, do not work anymore with 4.0.

To pre-create users, you should first create the Drupal users in your preferred way (either by using the administration interface or by direct SQL query), and then you **MUST** manually run the following **three queries**:

```
INSERT INTO shib_authmap (uid, targeted_id)
  SELECT uid, name
  FROM users
  WHERE uid > 1 AND (uid) NOT IN
    (SELECT uid
    FROM shib_authmap);

INSERT INTO authmap (uid, authname)
  SELECT uid, targeted_id
```

```
  FROM shib_authmap
  WHERE (uid) NOT IN
    (SELECT uid
    FROM authmap);

UPDATE authmap SET module = 'shib_auth'
WHERE (uid) IN
  (SELECT uid
  FROM shib_authmap);
```

Note

These queries add all your existing users to `shib_authmap` with their usernames and make sure that your `authmap` table contains all of the user entries. You should optimize these queries if you have a large number of users.

Figyelem

You should repeat these queries each time after you add pre-created users.

Role assignment

It's possible to assign roles to users based on their Shibboleth attributes.

An assignment rule is made of three parameters:

- **$_SERVER** header name: name of the Shibboleth-derived attribute
- **Value regexp**: regexp applied to (all) the value(s) of the Shibboleth-derived attribute
- **Role(s)**: checklist of roles to be assigned to the matching users

Dynamic rules (default)

Dynamic rules add roles to the user, but **do not save them to the user's profile**. This means that

- the roles assigned by dynamic rules are **NOT displayed on the user page**, even though the permissions assigned to the role are in effect
- the roles assigned by dynamic rules are **automatically revoked** if the Shibboleth attributes change (ie. `eduPersonAffiliation` changes from `student` to `alum`),
- thus if the user logs in by other means than Shibboleth, the rule will not be triggered, so the roles will not be assigned.

Additional roles can be assigned statically to the user (as an individual) by the administrator as normally. Every logged in user gets the role `Authenticated user` automatically.

Sticky rules

On the other hand, sticky rules do **save the roles to the user's profile**. Thus

- the roles assigned by sticky rules are displayed on the user page,
- the roles are not revoked automatically by the module,
- the roles will be in effect regardless of the login procedure.

User profile mapping

From the 7.x-4.2 version (D6 is not supported) it is possible to define a mapping between Shibboleth attributes and Drupal Fields. You must have the *Field UI* and the *Shibboleth profile fields* modules enabled to use this functionality. Unlike other features of the module, this mapping is configured together with the field definition.

Go to *Administration » Configuration » People » Account settings » Manage fields* (`admin/config/people/accounts/fields`) and create a new field or edit an existing one. The Shibboleth mapping is available on the Field Edit form and can be used in three ways:

- *Disabled*: no mapping (this is the default);
- *Initial value from Shibboleth, later editable by User*: the value of the mapping is only assigned to the field if the field has no values;
- *Always update value on User login, not editable by User*: the field is updated on every login.

You can use the values of the server variables by referring to them with square brackets like `[sn]`. You can reference multiple server variables in one mapping. Anything that is not matched to a server variable will be treated as string and copied to the value of the field. The server variable match is case insensitive.

As an example, consider the user's request containing the following server variables (regardless of being set by Shibboleth or by something else):

```
[givenName] -> John
[sn]        -> Doe
[email]     -> jdoe@example.com
```

The following mappings would produce the results as indicated:

`[sn], [givenName] <[email]>` ***Doe, John jdoe@example.com*** `[firstName] [sn]` ***[firstname] Doe*** (note the mistaken header name)

Account linking

There might be cases when you have a number of existing users and you want them to (optionally) log in through the federation. If you enable **account linking**, a user can add her SSO login to her existing Drupal account. The process of adding an SSO login -> Drupal account association is the following (all steps are performed by the user):

1. Login to Drupal (with username/password)
2. Go to `My account -> Edit`
3. Click on `Link this account to Shibboleth!`
4. Authenticate with your IdP

From this point the user can choose to login either with Shibboleth or with username/password. This feature can also be useful for users switching home organizations.

> **Info**
>
> • One Drupal account can be linked to more than one federated unique identifier, however
> • One federated identifier can only be linked to a single Drupal account
> • Nobody can link to *admin* or *Anonymous user*
> • If the administrator disables this feature, no new associations can be stored. Existing associations remain in effect.
> • On a new user logon, the one cannot choose the Drupal username of another user (when *user-defined usernames* is active). For account linking, the user must be already logged in.
> • Currently account linking link is not displayed if the user has been logged in using Shibboleth. If you want to support users with multiple federated identities, please file a feature request.

> **Figyelem**
>
> Dynamic roles are roles based on server variables, not users. These may well be different on username/password logon and Shibboleth logon.

Logging out

Session expiry

Enable the option "*Destroy Drupal session when the Shibboleth session expires*", if you want to force logout the users without a valid Shibboleth session. (This only applies to lazy sessions, otherwise it is the webserver what ensures that you have a valid session.)

>

Info

Keep in mind if you leave this option off:

- if the Shibboleth session is lost, assigned dynamic roles are lost, too
- Shibboleth session might get lost if you use a clustered SP without a central session cache
- **Never ever leave this option off if you want support Single Logout**. Otherwise the user will remain logged in to Drupal even after doing single (global) logout.

URL to redirect to after logout

Define an URL here, where you want the user to be navigated after logout. The URL can be absolute or relative to the server base url. The relative paths will be automatically extended with the site base URL.

SAML2 Logout

At the moment, Shibboleth2 SP supports SAML2 logout while the Shibboleth2 IdP does not. It has a consequence that (if you have a standard Shibboleth2 installation), you will get a Shibboleth error message on logout, like this:

```
Global Logout
Status of Global Logout: Identity provider does not support SAML 2 Single Logout protocol.
```

You can avoid this message by commenting out SAML2 global logout initiator from `/Logout` handler in `/etc/shibboleth/shibboleth2.xml`:

```
<!-- LogoutInitiators enable SP-initiated local or global/single logout of sessions. -->
<LogoutInitiator type="Chaining" Location="/Logout" relayState="cookie">
  <!-- The following line should be commented out to make Drupal logout work,
       as long as your IdPs do not support SAML2 logout -->
  <!--LogoutInitiator type="SAML2" template="bindingTemplate.html"/-->
  <LogoutInitiator type="Local"/>
</LogoutInitiator>
```

User consent

In certain scenarios you are only allowed to store personal data if the user accepted some kind of legal agreement such as the Terms of Use or the Privacy Policy.

When a new user arrives, she gets a screen with a link to the legal document and a checkbox. This page also contains the user's name and the e-mail address. Personal data required for login are stored only if the user accepts the agreement. If it's allowed by the administrator, the user might set custom values for those attributes.

If the document changes, the administrator might increment the version. This case all users are forced to accept the new version of the agreement before logging in.

> **Note**
>
> Version matching is an exact match, so the user must accept exactly the same version what was specified by the administrator

Personal data handled by the module

- username
- user's e-mail address
- user's targeted identifier(s) (what is sent by the IdP, if different from username), along with a mapping to the Drupal username
- Identity Provider(s) that identified the user
- random password (this might not be considered personal data)
- time of the registration

Advanced SAML2 features

SAML2 defines several properties of the authentication request message which may affect the authentication performed by the IdP.

> **Warning**
>
> **Be careful when using these options.**
>
> IdPs that do not support these features might signal an error instead of performing any kind of authentication of the user. Or might show errors *after* authenticating the user. Some kind of authentication handlers (ie. HTTP Basic Auth) will never work even if the IdP software is capable of handling these properties.

isPassive

If `isPassive` is set, then the user is redirected to the IdP or to the Discovery Service in advance, but neither of them are allowed to take over the control of the user interface (such as performing any visible authentication). If authentication (or IdP selection) can not be performed silently, an error is

returned, which is then handled by the module.

By using this option, you can auto-login users who are already logged in to the IdP even if you are using lazy sessions. If auto-login can not be performed, the user returns to Drupal unauthenticated without seeing any error message.

> **Figyelem**
>
> You need to instruct Shibboleth to redirect errors back to Drupal to handle passive authentication failures. This is done by setting `redirectErrors` parameter in the RequestMap. See Shibboleth wiki for details.
>
> **IMPORTANT**: current implementation does not handle any errors except for NoPassive error. This means, that on every possible Shibboleth SP error you will get an unauthenticated Drupal page instead of a Shibboleth error page.

> **Note**
>
> this feature requires JavaScript.

forceAuthn

This feature requires reauthentication of the user at the IdP even if she is having an SSO session there. Just like isPassive, it's not generally supported by authentication handlers.

In general, you should never mix **isPassive** and **forceAuthn**. The standard states that in this case, the IdP must reauthenticate the user without visibly taking control of the user interface. It's up to you, how you interpret it...

Non-implemented features

There are several other SAML features which are not yet implemented, because there seems to be very little practical interest for them. Please file a feature request if you really need them, but before doing so, please, spend a little time on thinking how things should really work.

- **AuthnContextClass**: the SP can request some specific authentication context from the IdP.
- **NameID management**: the IdP can request to change the user's NameID (targeted identifier) or to remove it.
- **Logout notification**: this is a Shibboleth2 feature, not a SAML one. You generally do not need this, because you can terminate the Drupal session on Shibboleth session expiry. This workaround has the disadvantage that the module will not be notified at the time when the user was logged out, only on the next page refresh.

Change log

Version 3.0 -> 3.1

If you need documentation for 3.0, please use the previous version of the documentation

Release notes

Version 4.0

Bug fixes

- The module now works with caching enabled
- Code refactoring

New features

- Allow specifying Drupal username, thus support opaque identifiers
- Support for sticky rules (roles saved permanently to the users' profile)
- Basic support for account linking
- Basic support for getting user consent on personal data
- Support for isPassive and forceAuthn session initiation
- Support for redirecting users to HTTPS on login

Változat #2
cziernorbert hozta létre 2026-04-14 13:22:45 CEST
cziernorbert frissítette 2026-04-17 12:58:12 CEST