Authentication Plugins

Stackable Authentication Method(s)

Since many institutions and organizations have existing authentication systems, DSpace has been designed to allow these to be easily integrated into an existing authentication infrastructure. It keeps a series, or "stack", of *authentication methods*, so each one can be tried in turn. This makes it easy to add new authentication methods or rearrange the order without changing any existing code. You can also share authentication code with other sites.

Configuration File:	**`[dspace]/config/modules/authentication.cfg`**
Property:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod`

Example Value:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod = \`
	`org.dspace.authenticate.PasswordAuthentication`

The configuration property `plugin.sequence.org.dspace.authenticate.AuthenticationMethod` defines the authentication stack. It is a comma-separated list of class names. Each of these classes implements a different `authentication method`, or way of determining the identity of the user. They are invoked in the order specified until one succeeds.

Existing Authentication Methods include

- Authentication by Password (class: `org.dspace.authenticate.PasswordAuthentication`) (DEFAULT)
- Shibboleth Authentication (class: `org.dspace.authenticate.ShibAuthentication`)
- LDAP Authentication (class: `org.dspace.authenticate.LDAPAuthentication`)
  - Hierarchical LDAP Authentication (class: `org.dspace.authenticate.LDAPHierarchicalAuthentication`)
- IP Address based Authentication (class: `org.dspace.authenticate.IPAuthentication`)
- X.509 Certificate Authentication (class: `org.dspace.authenticate.X509Authentication`)

An authentication method is a class that implements the interface `org.dspace.authenticate.AuthenticationMethod`. It `authenticates` a user by evaluating the *credentials* (e.g. username and password) he or she presents and checking that they are valid.

The basic authentication procedure in the DSpace Web UI is this:

1. A request is received from an end-user's browser that, if fulfilled, would lead to an action requiring authorization taking place.
2. If the end-user is already authenticated:
   - If the end-user is allowed to perform the action, the action proceeds
   - If the end-user is NOT allowed to perform the action, an authorization error is displayed.
   - If the end-user is NOT authenticated, i.e. is accessing DSpace anonymously:
3. The parameters etc. of the request are stored.
4. The Web UI's `startAuthentication` method is invoked.
5. First it tries all the authentication methods which do `implicit` authentication (i.e. they work with just the information already in the Web request, such as an X.509 client certificate). If one of these succeeds, it proceeds from Step 2 above.
6. If none of the implicit methods succeed, the UI responds by putting up a "login" page to collect credentials for one of the `explicit` authentication methods in the stack. The servlet processing that page then gives the proffered credentials to each authentication method in turn until one succeeds, at which point it retries the original operation from Step 2 above.
   Please see the source files `AuthenticationManager.java` and `AuthenticationMethod.java` for more details about this mechanism.

Authentication by Password

Enabling Authentication by Password

By default, this authentication method is enabled in DSpace.

However, to enable Authentication by Password, you must ensure the `org.dspace.authenticate.PasswordAuthentication` class is listed as one of the AuthenticationMethods in the following configuration:

Configuration File:	**`[dspace]/config/modules/authentication.cfg`**
Property:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod`

Example Value:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod = \`
	` org.dspace.authenticate.PasswordAuthentication`

Configuring Authentication by Password

The default method `org.dspace.authenticate.PasswordAuthentication` has the following properties:

- Use of inbuilt e-mail address/password-based log-in. This is achieved by forwarding a request that is attempting an action requiring authorization to the password log-in servlet, `/password-login`. The password log-in servlet (`org.dspace.app.webui.servlet.PasswordServlet`) contains code that will resume the original request if authentication is successful, as per step 3. described above.
- Users can register themselves (i.e. add themselves as e-people without needing approval from the administrators), and can set their own passwords when they do this
- Users are not members of any special (dynamic) e-person groups
- You can restrict the domains from which new users are able to register. To enable this feature, uncomment the following line from dspace.cfg: `authentication.password.domain.valid = example.com` Example options might be '`@example.com`' to restrict registration to users with addresses ending in @example.com, or '`@example.com, .ac.uk`' to restrict registration to users with addresses ending in @example.com or with addresses in the .ac.uk domain.

A full list of all available Password Authentication Configurations:

Configuration File:	**`[dspace]/config/modules/authentication-password.cfg`**
Property:	`domain.valid`
Example Value:	`domain.value = @mit.edu, .ac.uk`
Informational Note:	This option allows you to limit self-registration to email addresses ending in a particular domain value. The above example would limit self-registration to individuals with "@mit.edu" email addresses and all ".ac.uk" email addresses.
Property:	`login.specialgroup`
Example Value:	`login.specialgroup = My DSpace Group`
Informational Note:	This option allows you automatically add all password authenticated users to a specific DSpace Group (the group must exist in DSpace) for the remainder of their logged in session.

Shibboleth Authentication

Enabling Shibboleth Authentication

To enable Shibboleth Authentication, you must ensure the `org.dspace.authenticate.ShibAuthentication` class is listed as one of the AuthenticationMethods in the following configuration:

Configuration File:	**`[dspace]/config/modules/authentication.cfg`**
Property:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod`

Example Value:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod = \`
	` org.dspace.authenticate.ShibAuthentication`

Configuring Shibboleth Authentication (DSpace 1.8.0)

Additional Instructions

Detailed instructions for installing Shibboleth on DSpace 1.5.x may be found at https://mams.melcoe.mq.edu.au/zope/mams/pubs/Installation/dspace15.

Once it has been enabled (see above), Shibboleth Authentication is configured via its own `[dspace]/config/modules/authentication-shibboleth.cfg` file.

DSpace requires an email address as the user's credentials. There are two ways of providing email to DSpace from Shibboleth:

1. By explicitly specifying to the user which attribute (header) carries the email address.
2. By turning on the `user-email-using-tomcat=true` which means the software will attempt to acquire the user's email from Tomcat.

The first option takes **Precedence** when specified. both options can be enabled to allow for fallback.

A full list of all available Shibboleth Configurations:

Configuration File:	**`[dspace]/config/modules/authentication-shibboleth.cfg`**
Property:	`email-header`
Example Value:	`email-header = MAIL`
Informational Note:	The option specifies that the email comes from the mentioned header. This value is CASE-Sensitive.
Property:	`firstname-header`
Example Value:	`firstname-header = SHIB-EP-GIVENNAME`
Informational Note:	Optional. Specify the header that carries the user's first name. This is going to be used for the creation of new-user.
Property:	`lastname-header`
Example Value:	`lastname-header = SHIB-EP-SURNAME`
Informational Note:	Optional. Specify the header that carries user's last name. This is used for creation of new user.
Property:	`email-use-tomcat-remote-user`
Example Value:	`email-use-tomcat-remote-user = true`
Informational Note:	This option forces the software to acquire the email from Tomcat.
Property:	`autoregister`
Example Value:	`autoregister = true`
Informational Note:	Option will allow new users to be registered automatically if the IdP provides sufficient information (and the user does not exist in DSpace).
Property:	`role-header` `role-header.ignore-scope`

Example Value:	`role-header = Shib-EP-ScopedAffiliation`
	`role-header.ignore-scope = true`
	or
	`role-header = Shib-EP-UnscopedAffiliation`
	`role-header.ignore-scope = false`

Informational Note:	These two options specify which attribute that is responsible for providing user's roles to DSpace and unscope the attributes if needed. When not specified, it is defaulted to 'Shib-EP-UnscopedAffiliation', and ignore-scope is defaulted to 'false'. The value is specified in *AAP.xml* (Shib 1.3.x) or *attribute-filter.xml* (Shib 2.x). The value is CASE-Sensitive. The values provided in this header are separated by semi-colon or comma. If your service provider (SP) only provides scoped role header, you need to set `role-header.ignore-Scope` as '*true*'. For example if you only get Shib-EP-ScopedAffiliation instead of Shib-EP-ScopedAffiliation, you name to make your settings as in the example value above.

Property:	`default-roles`
Example Value:	`default-roles = Staff, Walk-ins`
Informational Note:	When user is fully authN or IdP but would not like to release his/her roles to DSpace (for privacy reasons?), what should the default roles be given to such user. The values are separated by semi-colon or comma.

Property:	`role.Senior\ Researcher`
	`role.Librarian`

Example Value:	`role.Senior\ Researcher = Researcher, Staff`
	`role.Librarian = Administrator`

Informational Note:	The following mappings specify role mapping between IdP and Dspace. The left side of the entry is IdP's role (prefixed with 'role.') which will be mapped to the right entry from DSpace. DSpace's group as indicated on the right entry has to EXIST in DSpace, otherwise user will be identified as 'anonymous'. Multiple values on the right entry should be separated by comma. The values are CASE-Sensitive. Heuristic one-to-one mapping will be done when the IdP groups entry are not listed below (i.e. if 'X' group in IdP is not specified here, then it will be mapped to 'X' group in DSpace if it exists, otherwise it will be mapped to simply 'anonymous'). Given sufficient demand, future release could support regex for the mapping special characters need to be escaped by '\'

Configuring Shibboleth Authentication (DSpace 1.8.1)

Important Note

Because of a mix-up with the 1.8.1 release the configuration parameters for the Shibboleth plugin reverted to their previous 1.7.x style. These parameters should be included in your dspace.cfg instead of the modules directory for this release. Future releases will fix this and they will be moved back to the config modules style.

Shibboleth is a distributed authentication system for securely authenticating users and passing attributes about the user from one or more identity providers. In the Shibboleth terminology DSpace is a Service Provider which receives authentication information and then based upon that provides a service to the user. With Shibboleth DSpace will require that you use Apache installed with the mod_shib module acting as a proxy for all HTTP requests for your servlet container (typically Tomcat). DSpace will receive authentication information from the mod_shib module through HTTP headers.

See for more information on installing and configuring a Shibboleth Service Provider: https://wiki.shibboleth.net/confluence/display/SHIB2/Installation

Sessions:

When configuring your Shibboleth Service Provider there are two paradigms you may use: Active or Lazy Sessions. Active sessions is where the mod_shib module is configured to product a URL space. No one will be able to access that URL without first authenticating with Shibboleth. Using this method you will need to configure shibboleth to protect the URL: "/shibboleth-login". The alternative, Lazy Session does not protect any specific URL. Instead apache will allow access to any URL, and when the application wants to it may initiate an authenticated session. The Lazy Session method is preferable for most DSpace installations where you want public access to most of DSpace but restricted access to limited areas - such as administration.

Authentication Methods:

DSpace supports authentication using NetID, or email address. A user's NetID is a unique identifier from the IdP that identifies a particular user. The NetID can be of almost any form such as a unique integer, string, or with Shibboleth 2.0 you can use "targeted ids". You will need to coordinate with your shibboleth federation or identity provider. There are three ways to supply identity information to DSpace:

*1) NetID from Shibboleth Header (**best**)*

The NetID-based method is superior because users may change their email address with the identity provider. When this happens DSpace will not be able to associate their new address with their old account.

*2) Email address from Shibboleth Header (**okay**)*

In the case where a NetID header is not available or not found DSpace will fall back to identifying a user based-upon their email address.

*3) Tomcat's Remote User (**worst**)*

In the event that neither Shibboleth headers are found then as a last resort DSpace will look at Tomcat's remote user field. This is the least attractive option because Tomcat has no way to supply additional attributes about a user. Because of this the autoregister option is not supported if this method is used.

Identity Scheme Migration Strategies:

If you are currently using Email based authentication (either 1 or 2) and want to upgrade to NetID based authentication then there is an easy path. Simply enable shibboleth to pass the NetID attribute and set the netid-header below to the correct value. When a user attempts to log in to DSpace first DSpace will look for an EPerson with the passed NetID, however when this fails DSpace will fall back to email based authentication. Then DSpace will update the user's EPerson account record to set their netted so all future authentications for this user will be based upon netted. One thing to note is that DSpace will prevent an account from switching NetIDs. If an account all ready has a NetID set and then they try and authenticate with a different NetID the authentication will fail.

EPerson Metadata:

One of the primary benefits of using Shibboleth based authentication is receiving additional attributes about users such as their names, telephone numbers, and possibly their academic department or graduation semester if desired. DSpace treats the first and last name attributes differently because they (along with email address) are the three pieces of minimal information required to create a new user account. For both first and last name supply direct mappings to the Shibboleth headers. In additional to the first and last name DSpace supports other metadata fields such as phone, or really anything you want to store on an eperson object. Beyond the phone field, which is accessible in the user's profile screen, none of these additional metadata fields will be used by DSpace out-of-the box. However if you develop any local modification you may access these attributes from the EPerson object. The Vireo ETD workflow system utilizes this to aid students when submitting an ETD.

Role-based Groups:

DSpace is able to place users into pre-defined groups based upon values received from Shibboleth. Using this option you can place all faculty members into a DSpace group when the correct affiliation's attribute is provided. When DSpace does this they are considered 'special groups', these are really groups but the user's membership within these groups is not recorded in the database. Each time a user authenticates they are automatically placed within the pre-defined DSpace group, so if the user loses their affiliation then the next time they login they will no longer be in the group.

Depending upon the shibboleth attributed use in the role-header it may be scoped. Scoped is shibboleth terminology for identifying where an attribute originated from. For example a students affiliation may be encoded as "student@tamu.edu". The part after the @ sign is the scope, and the preceding value is the value. You may use the whole value or only the value or scope. Using this you could generate a role for students and one institution different than students at another institution. Or if you turn on ignore-scope you could ignore the institution and place all students into one group.

The values extracted (a user may have multiple roles) will be used to look up which groups to place the user into. The groups are defined as "authentication.shib.role.<role-name>" which is a comma separated list of DSpace groups.

Configuration File:	**`[dspace]/config/dspace.cfg`**
Property:	`authentication.shib.lazysession`
Example Value:	`authentication.shib.lazysession = true`
Informational Note:	Whether to use lazy sessions or active sessions.
Property:	`authentication.shib.lazysession.loginurl`
Example Value:	`authentication.shib.lazysession.loginurl = /Shibboleth.sso/Login`
Informational Note:	The url to start a shibboleth session (only for lazy sessions)
Property:	`authentication.shib.lazysession.secure`
Example Value:	`authentication.shib.lazysession.secure = true`
Informational Note:	Force HTTPS when authenticating (only for lazy sessions)
Property:	`authentication.shib.netid-header`
Example Value:	`authentication.shib.netid-header = SHIB-NETID`
Informational Note:	The HTTP header where shibboleth will supply a user's NetID.
Property:	`authentication.shib.email-header`
Example Value:	`authentication.shib.email-header = SHIB-MAIL`
Informational Note:	The HTTP header where the shibboleth will supply a user's email address.
Property:	`authentication.shib.email-use-tomcat-remote-user`
Example Value:	`authentication.shib.email-use-tomcat-remote-user = false`
Informational Note:	Used when a netid or email heades are not available should shibboleth authentication fall back to using Tomcat's remote user feature.
Property:	`authentication.shib.autoregister`
Example Value:	`authentication.shib.autoregister = true`
Informational Note:	Should we allow new users to be registered automatically?

Property:	`authentication.shib.sword.compatability`
Example Value:	`authentication.shib.sword.compatability = true`
Informational Note:	Sword compatability will allow this authentication method to work when using sword. Sort relies on username and password based authentication and is entirely incapable of supporting shibboleth. This option allows you to authenticate username and passwords for sword sessions with out adding another authentication method onto the stack. You will need to ensure that a user has a password. One way to do that is to create the user via the create-administrator command line command and then edit their permissions.
Property:	`authentication.shib.firstname-header`
Example Value:	`authentication.shib.firstname-header = SHIB_GIVENNAME`
Informational Note:	The HTTP header where the shibboleth will supply a user's given name.
Property:	`authentication.shib.lastname-header`
Example Value:	`authentication.shib.lastname-header = SHIB_SN`
Informational Note:	The HTTP header where the shibboleth will supply a user's sur name.
Property:	`authentication.shib.eperson.metadata`

Example Value:	`authentication.shib.eperson.metadata = \`
	`SHIB-telephone => phone, \`
	`SHIB-cn => cn`

Informational Note:	Additional user attributes mapping, multiple attributes may be stored for each user. The left side is the Shibboleth-based metadata Header and the right side is the eperson metadata field to map the attribute to.
Property:	`authentication.shib.eperson.metadata.autocreate`
Example Value:	`authentication.shib.eperson.metadata.autocreate = true`
Informational Note:	If the eperson metadata field is not found, should it be automatically created?
Property:	`authentication.shib.role-header`
Example Value:	`authentication.shib.role-header = SHIB_SCOPED_AFFILIATION`
Informational Note:	The shibboleth header to do role-based mappings (see section on roll based mapping section above)
Property:	`authentication.shib.role-header.ignore-scope`
Example Value:	`authentication.shib.role-header.ignore-scope = true`
Informational Note:	Weather to ignore the attribute's scope (everything after the @ sign for scoped attributes)
Property:	`authentication.shib.role-header.ignore-value`
Example Value:	`authentication.shib.role-header.ignore-value = false`
Informational Note:	Weather to ignore the attribute's value (everything before the @ sign for scoped attributes)
Property:	`authentication.shib.role.[affiliation-attribute]`

Example Value:	`authentication.shib.role.faculty = Faculty, Member \`
	`authentication.shib.role.staff = Staff, Member \`
	`authentication.shib.role.student = Students, Member`

Informational Note:	Mapping of affiliation values to DSpace groups.(See the roll based mapping section above)

Configuring Shibboleth Authentication (DSpace 1.8.2+)

Shibboleth is a distributed authentication system for securely authenticating users and passing attributes about the user from one or more identity providers. In the Shibboleth terminology DSpace is a Service Provider which receives authentication information and then based upon that provides a service to the user. With Shibboleth DSpace will require that you use Apache installed with the mod_shib module acting as a proxy for all HTTP requests for your servlet container (typically Tomcat). DSpace will receive authentication information from the mod_shib module through HTTP headers.

See for more information on installing and configuring a Shibboleth Service Provider: https://wiki.shibboleth.net/confluence/display/SHIB2/Installation

Sessions:

When configuring your Shibboleth Service Provider there are two paradigms you may use: Active or Lazy Sessions. Active sessions is where the mod_shib module is configured to product a URL space. No one will be able to access that URL without first authenticating with Shibboleth. Using this method you will need to configure shibboleth to protect the URL: "/shibboleth-login". The alternative, Lazy Session does not protect any specific URL. Instead apache will allow access to any URL, and when the application wants to it may initiate an authenticated session. The Lazy Session method is preferable for most DSpace installations where you want public access to most of DSpace but restricted access to limited areas - such as administration.

Authentication Methods:

DSpace supports authentication using NetID, or email address. A user's NetID is a unique identifier from the IdP that identifies a particular user. The NetID can be of almost any form such as a unique integer, string, or with Shibboleth 2.0 you can use "targeted ids". You will need to coordinate with your shibboleth federation or identity provider. There are three ways to supply identity information to DSpace:

*1) NetID from Shibboleth Header (**best**)*

The NetID-based method is superior because users may change their email address with the identity provider. When this happens DSpace will not be able to associate their new address with their old account.

*2) Email address from Shibboleth Header (**okay**)*

In the case where a NetID header is not available or not found DSpace will fall back to identifying a user based-upon their email address.

*3) Tomcat's Remote User (**worst**)*

In the event that neither Shibboleth headers are found then as a last resort DSpace will look at Tomcat's remote user field. This is the least attractive option because Tomcat has no way to supply additional attributes about a user. Because of this the autoregister option is not supported if this method is used.

Identity Scheme Migration Strategies:

If you are currently using Email based authentication (either 1 or 2) and want to upgrade to NetID based authentication then there is an easy path. Simply enable shibboleth to pass the NetID attribute and set the netid-header below to the correct value. When a user attempts to log in to DSpace first DSpace will look for an EPerson with the passed NetID, however when this fails DSpace will fall back to email based authentication. Then DSpace will update the user's EPerson account record to set their netted so all future authentications for this user will be based upon netted. One thing to note is that DSpace will prevent an account from switching NetIDs. If an account all ready has a NetID set and then they try and authenticate with a different NetID the authentication will fail.

EPerson Metadata:

One of the primary benefits of using Shibboleth based authentication is receiving additional attributes about users such as their names, telephone numbers, and possibly their academic department or graduation semester if desired. DSpace treats the first and last name attributes differently because they (along with email address) are the three pieces of minimal information required to create a new user account. For both first and last name supply direct mappings to the Shibboleth headers. In additional to the first and last name DSpace supports other metadata fields such as phone, or really anything you want to store on an eperson object. Beyond the phone field, which is accessible in the user's profile screen, none of these additional metadata fields will be used by DSpace out-of-the box. However if you develop any local modification you may access these attributes from the EPerson object. The Vireo ETD workflow system utilizes this to aid students when submitting an ETD.

Role-based Groups:

DSpace is able to place users into pre-defined groups based upon values received from Shibboleth. Using this option you can place all faculty members into a DSpace group when the correct affiliation's attribute is provided. When DSpace does this they are considered 'special groups', these are really groups but the user's membership within these groups is not recorded in the database. Each time a user authenticates they are automatically placed within the pre-defined DSpace group, so if the user loses their affiliation then the next time they login they will no longer be in the group.

Depending upon the shibboleth attributed use in the role-header it may be scoped. Scoped is shibboleth terminology for identifying where an attribute originated from. For example a students affiliation may be encoded as "student@tamu.edu". The part after the @ sign is the scope, and the preceding value is the value. You may use the whole value or only the value or scope. Using this you could generate a role for students and one institution different than students at another institution. Or if you turn on ignore-scope you could ignore the institution and place all students into one group.

The values extracted (a user may have multiple roles) will be used to look up which groups to place the user into. The groups are defined as "role.<role-name>" which is a comma separated list of DSpace groups.

Configuration File:	`[dspace]/config/modules/authentication-shibboleth.cfg`
Property:	`lazysession`

Example Value:	`lazysession = true`
Informational Note:	Whether to use lazy sessions or active sessions.
Property:	`lazysession.loginurl`
Example Value:	`lazysession.loginurl = /Shibboleth.sso/Login`
Informational Note:	The url to start a shibboleth session (only for lazy sessions)
Property:	`lazysession.secure`
Example Value:	`lazysession.secure = true`
Informational Note:	Force HTTPS when authenticating (only for lazy sessions)
Property:	`netid-header`
Example Value:	`netid-header = SHIB-NETID`
Informational Note:	The HTTP header where shibboleth will supply a user's NetID.
Property:	`email-header`
Example Value:	`email-header = SHIB-MAIL`
Informational Note:	The HTTP header where the shibboleth will supply a user's email address.
Property:	`email-use-tomcat-remote-user`
Example Value:	`email-use-tomcat-remote-user = false`
Informational Note:	Used when a netid or email heades are not available should shibboleth authentication fall back to using Tomcat's remote user feature.
Property:	`autoregister`
Example Value:	`autoregister = true`
Informational Note:	Should we allow new users to be registered automatically?
Property:	`sword.compatability`
Example Value:	`sword.compatability = true`
Informational Note:	Sword compatability will allow this authentication method to work when using sword. Sort relies on username and password based authentication and is entirely incapable of supporting shibboleth. This option allows you to authenticate username and passwords for sword sessions with out adding another authentication method onto the stack. You will need to ensure that a user has a password. One way to do that is to create the user via the create-administrator command line command and then edit their permissions.
Property:	`firstname-header`
Example Value:	`firstname-header = SHIB_GIVENNAME`
Informational Note:	The HTTP header where the shibboleth will supply a user's given name.
Property:	`lastname-header`
Example Value:	`lastname-header = SHIB_SN`
Informational Note:	The HTTP header where the shibboleth will supply a user's sur name.
Property:	`eperson.metadata`

Example Value:	`eperson.metadata = \`
	` SHIB-telephone => phone, \`
	` SHIB-cn => cn`

Informational Note:	Additional user attributes mapping, multiple attributes may be stored for each user. The left side is the Shibboleth-based metadata Header and the right side is the eperson metadata field to map the attribute to.
Property:	`eperson.metadata.autocreate`
Example Value:	`eperson.metadata.autocreate = true`
Informational Note:	If the eperson metadata field is not found, should it be automatically created?
Property:	`role-header`
Example Value:	`role-header = SHIB_SCOPED_AFFILIATION`
Informational Note:	The shibboleth header to do role-based mappings (see section on roll based mapping section above)
Property:	`role-header.ignore-scope`
Example Value:	`role-header.ignore-scope = true`
Informational Note:	Weather to ignore the attribute's scope (everything after the @ sign for scoped attributes)
Property:	`role-header.ignore-value`
Example Value:	`role-header.ignore-value = false`
Informational Note:	Weather to ignore the attribute's value (everything before the @ sign for scoped attributes)
Property:	`role.[affiliation-attribute]`

Example Value:	`role.faculty = Faculty, Member \`
	` role.staff = Staff, Member \`
	` role.student = Students, Member`

Informational Note:	Mapping of affiliation values to DSpace groups.(See the roll based mapping section above)

LDAP Authentication

Enabling LDAP Authentication

To enable LDAP Authentication, you must ensure the `org.dspace.authenticate.LDAPAuthentication` class is listed as one of the AuthenticationMethods in the following configuration:

Configuration File:	**`[dspace]/config/modules/authentication.cfg`**
Property:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod`

Example Value:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod = \`
	` org.dspace.authenticate.LDAPAuthentication`

Configuring LDAP Authentication

If LDAP is enabled, then new users will be able to register by entering their username and password without being sent the registration token. If users do not have a username and password, then they can still register and login with just their email address the same way they do now.

If you want to give any special privileges to LDAP users, create a stackable authentication method to automatically put people who have a netid into a special group. You might also want to give certain email addresses special privileges. Refer to the Custom Authentication Code section below for more information about how to do this.

Here is an explanation of each of the different LDAP configuration parameters:

Configuration File:	**`[dspace]/config/modules/authentication-ldap.cfg`**
Property:	`enable`
Example Value:	`enable = false`
Informational Note:	This setting will enable or disable LDAP authentication in DSpace. With the setting off, users will be required to register and login with their email address. With this setting on, users will be able to login and register with their LDAP user ids and passwords.
Property:	`autoregister`
Example Value:	`autoregister = true`
Informational Note:	This will turn LDAP autoregistration on or off. With this on, a new EPerson object will be created for any user who successfully authenticates against the LDAP server when they first login. With this setting off, the user must first register to get an EPerson object by entering their ldap username and password and filling out the forms.
Property:	`provider_url`
Example Value:	`provider_url = ldap://ldap.myu.edu/o=myu.edu`
Informational Note:	This is the url to your institution's LDAP server. You may or may not need the /o=myu.edu part at the end. Your server may also require the ldaps:// protocol.
Property:	`id_field`
Example Value:	`id_field = uid`
Explanation:	This is the unique identifier field in the LDAP directory where the username is stored.
Property:	`object_context`
Example Value:	`object_context = ou=people, o=myu.edu`
Informational Note:	This is the object context used when authenticating the user. It is appended to the *id_field* and username. For example `uid=username,ou=people,o=myu.edu`. You will need to modify this to match your LDAP configuration.
Property:	`search_context`
Example Value:	`search_context = ou=people`
Informational Note:	This is the search context used when looking up a user's LDAP object to retrieve their data for autoregistering. With `autoregister` turned on, when a user authenticates without an EPerson object we search the LDAP directory to get their name and email address so that we can create one for them. So after we have authenticated against *uid=username,ou=people,o=byu.edu* we now search in ou=people for filtering on [uid=username]. Often the `search_context` is the same as the `object_context` parameter. But again this depends on your LDAP server configuration.
Property:	`email_field`
Example Value:	`email_field = mail`
Informational Note:	This is the LDAP object field where the user's email address is stored. "mail" is the default and the most common for LDAP servers. If the mail field is not found the username will be used as the email address when creating the eperson object.
Property:	`surname_field`
Example Value:	`surname_field = sn`
Informational Note:	This is the LDAP object field where the user's last name is stored. "sn" is the default and is the most common for LDAP servers. If the field is not found the field will be left blank in the new eperson object.
Property:	`givenname_field`
Example Value:	`givenname_field = givenName`
Informational Note:	This is the LDAP object field where the user's given names are stored. I'm not sure how common the givenName field is in different LDAP instances. If the field is not found the field will be left blank in the new eperson object.

Property:	`phone_field`
Example Value:	`phone_field = telephoneNumber`
Informational Note:	This is the field where the user's phone number is stored in the LDAP directory. If the field is not found the field will be left blank in the new eperson object.
Property:	`login.specialgroup`
Example Value:	`login.specialgroup = group-name`
Informational Note:	If required, a group name can be given here, and all users who log into LDAP will automatically become members of this group. This is useful if you want a group made up of all internal authenticated users. (Remember to log on as the administrator, add this to the "Groups" with read rights).

Enabling Hierarchical LDAP Authentication

If your users are spread out across a hierarchical tree on your LDAP server, you may wish to instead use the Hierarchical LDAP Authentication plugin.

To enable Hierarchical LDAP Authentication, you must ensure the `org.dspace.authenticate.LDAPHierarchicalAuthentication` class is listed as one of the AuthenticationMethods in the following configuration:

Configuration File:	**`[dspace]/config/modules/authentication.cfg`**
Property:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod`

Example Value:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod = \`
	` org.dspace.authenticate.LDAPHierarchicalAuthentication`

Configuring Hierarchical LDAP Authentication

Hierarchical LDAP Authentication shares all the above standard LDAP configurations, but has some additional settings.

You can optionally specify the search scope. If anonymous access is not enabled on your LDAP server, you will need to specify the full DN and password of a user that is allowed to bind in order to search for the users.

Configuration File:	**`[dspace]/config/modules/authentication-ldap.cfg`**
Property:	`search_scope`
Example Value:	`search_scope = 2`

Informational Note:	This is the search scope value for the LDAP search during autoregistering. This will depend on your LDAP server setup. This value must be one of the following integers corresponding to the following values:
	`object scope : 0`
	`one level scope : 1`
	`subtree scope : 2`

Property:	`search.user` `search.password`

Example Value:	`search.user = cn=admin,ou=people,o=myu.edu`
	`search.password = password`

Informational Note:	The full DN and password of a user allowed to connect to the LDAP server and search for the DN of the user trying to log in. If these are not specified, the initial bind will be performed anonymously.
Property:	`netid_email_domain`
Example Value:	`netid_email_domain = @example.com`
Informational Note:	If your LDAP server does not hold an email address for a user, you can use the following field to specify your email domain. This value is appended to the netid in order to make an email address. E.g. a netid of 'user' and `netid_email_domain` as `@example.com` would set the email of the user to be `user@example.com`

IP Authentication

Enabling IP Authentication

To enable IP Authentication, you must ensure the `org.dspace.authenticate.IPAuthentication` class is listed as one of the AuthenticationMethods in the following configuration:

Configuration File:	**`[dspace]/config/modules/authentication.cfg`**
Property:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod`

Example Value:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod = \`
	`org.dspace.authenticate.IPAuthentication`

Configuring IP Authentication

Configuration File:	**`[dspace]/config/modules/authentication-ip.cfg`**

Once enabled, you are then able to map DSpace groups to IP addresses in `authentication-ip.cfg` by setting `ip.GROUPNAME = iprange[, iprange ...]`, e.g:

```
ip.MY_UNIVERSITY = 10.1.2.3, \                  # Full IP
13.5, \                        # Partial IP
11.3.4.5/24, \                 # with CIDR
12.7.8.9/255.255.128.0, \      # with netmask
2001:18e8::32                  # IPv6 too
```

Negative matches can be set by prepending the entry with a '-'. For example if you want to include all of a class B network except for users of a contained class c network, you could use: 111.222,-111.222.333.

Notes:

- If the Groupname contains blanks you must escape the spaces, e.g. "Department\ of\ Statistics"
- If your DSpace installation is hidden behind a web proxy, remember to set the `useProxies` configuration option within the 'Logging' section of `dspace.cfg` to use the IP address of the user rather than the IP address of the proxy server.

X.509 Certificate Authentication

Enabling X.509 Certificate Authentication

The X.509 authentication method uses an X.509 certificate sent by the client to establish his/her identity. It requires the client to have a personal Web certificate installed on their browser (or other client software) which is issued by a Certifying Authority (CA) recognized by the web server.

1. See the HTTPS installation instructions to configure your Web server. If you are using HTTPS with Tomcat, note that the `<Connector>` tag *must* include the attribute `clientAuth="true"` so the server requests a personal Web certificate from the client.
2. Add the `org.dspace.authenticate.X509Authentication` plugin `first` to the list of stackable authentication methods in the value of the configuration key `plugin.sequence.org.dspace.authenticate.AuthenticationMethod`

Configuration File:	**`[dspace]/config/modules/authentication.cfg`**
Property:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod`

Example Value:	`plugin.sequence.org.dspace.authenticate.AuthenticationMethod = \`
	`org.dspace.authenticate.X509Authentication, \`
	`org.dspace.authenticate.PasswordAuthentication`

Configuring X.509 Certificate Authentication

Configuration File:	**`[dspace]/config/modules/authentication-x509.cfg`**

1. You must also configure DSpace with the same CA certificates as the web server, so it can accept and interpret the clients' certificates. It can share the same keystore file as the web server, or a separate one, or a CA certificate in a file by itself. Configure it by *one* of these methods, either the Java keystore

```
keystore.path =  path to Java keystore file
keystore.password =  password to access the keystore
```

...or the separate CA certificate file (in PEM or DER format):

```
ca.cert =  path to certificate file for CA whose client certs to accept.
```

2. Choose whether to enable auto-registration: If you want users who authenticate successfully to be automatically registered as new E-Persons if they are not already, set the `autoregister` configuration property to `true`. This lets you automatically accept all users with valid personal certificates. The default is `false`.

Example of a Custom Authentication Method

Also included in the source is an implementation of an authentication method used at MIT, *edu.mit.dspace.MITSpecialGroup*. This does not actually authenticate a user, it *only* adds the current user to a special (dynamic) group called 'MIT Users' (which must be present in the system!). This allows us to create authorization policies for MIT users without having to manually maintain membership of the MIT users group.

By keeping this code in a separate method, we can customize the authentication process for MIT by simply adding it to the stack in the DSpace configuration. None of the code has to be touched.

You can create your own custom authentication method and add it to the stack. Use the most similar existing method as a model, e.g. `org.dspace.authenticate.PasswordAuthentication` for an "explicit" method (with credentials entered interactively) or `org.dspace.authenticate.X509Authentication` for an implicit method.