IdPAuthUserPass

Configuring the IdP for Username/Password Authentication

This authentication handler supports authenticating users with a username (netid) and password. This authentication is performed through the Java Authentication and Authorization Service (JAAS). Prior to starting deployers should read JAAS Login Configuration File documentation as these instructions do not try to explain all the features available in this file.

JAAS configuration files are loaded into the VM's runtime configuration. Because of this, changes to the login configuration file are NOT reloaded if you stop and restart the IdP web application. You MUST restart the entire web application server.

Defining the Login Handler

This login handler is defined (in *$IDP_HOME/conf/handler.xml*) with the element `<LoginHandler xsi:type="UsernamePassword">` with the following required attribute:

- **jaasConfigurationLocation** - the JAAS configuration file to load, default: *$IDP_HOME/conf/login.config*

and the following optional attributes:

- **authenticationDuration** - length of time in minutes that the authentication method associated with this login handler is active; default: 30 minutes
- **authenticationServletURL** - context-relative path to the servlet responsible for collecting using credentials and authenticating the user; default: */Authn/UserPassword*

Additionally the login handler **must** contain one or more `<AuthenticationMethod>` elements whose content is the authentication method(s) serviced by the login handler.

Finally, you may also need to configure the Servlet with an "init" parameter in `web.xml` named `authnMethod`, set to a an authentication context/method /type value to return via SAML to the SP. By default, the value returned will be `urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport`. This is usually fine, but if you're doing something more advanced, it may need to be changed.

Note that if you were to assign the login handler to multiple `<AuthenticationMethod>` values up front, you will potentially have a problem because the Servlet itself can only return one of them. This may work fine, but would break if you are supporting SAML 2.0 SPs that request particular methods. You will probably need a custom login handler or handlers in such cases.

The Login Page

The **example** login page is located, in the IdP distribution, at *src/main/webapp/login.jsp*. For information on how to customize the login page see the login page customization document.

Setting up JAAS

The JAAS authentication policy to use may be specified in a JRE/container specific manner or by way to the *jaasConfigurationLocation* attribute on the *AuthenticationHandler* element as mentioned above. Deployers wishing to use a JRE or container specific manner must refer to the documentation for that specific method.

Shibboleth requires deployers to use **ShibUserPassAuth** as their JAAS policy application configuration entry.

Logging can be a bit of problem with JAAS because it is part of the VM itself (and as far as we can tell contains no logging messages). When possible we try to provide guidance on which package to enable logging for.

LDAP Login Module Specification

You should also check out the collected idiosyncrasies of different LDAP products that have been found by deployers.

Most LDAP directories treat the user ID as case insensitive for authentication purposes. The IdP is however case-preserving throughout its code, so while the directory may treat "jsmith" and "JSMITH" as the same user, the IdP will preserve the case differences. Please be aware of this if you attempt to use the principal name within any attribute definition or filter rules.

Shibboleth ships with an LDAP authentication module, `edu.vt.middleware.ldap.jaas.LdapLoginModule` that supports the follow properties.

Property Name	Property Description

ldapUrl	Fully qualified URL of the LDAP server, ldap://my.ldap.com:389. Supports a space separated list where each host is tried until a connection can be made.
baseDn	Search base for the LDAP server
bindDn	DN of privileged user used to connect to the LDAP server
bindCredential	Password for the privileged user
ssl	Whether to use SSL when connecting to the server, acceptable values are "true" or "false", default value is "false"
tls	Whether to use StartTLS when connecting to the server, acceptable values are "true" or "false", default value is "false"
userFilter	LDAP filter used to search for the user `uid={0}`
subtreeSearch	Whether to perform the search against all subtrees of the search base DN specified, or just the immediate children of the base DN, acceptable values are "true" or "false", default value is "false"
sslSocketFactory	fully qualified class name which implements javax.net.ssl.SSLSocketFactory. Can also be used to configure SSL trust: {trustCertificates=file:/path/to/trust.pem Use of SSL or TLS requires a valid signature chain from the LDAP server's certificate to a trusted CA. If using a self-signed cert on the LDAP server, it can be imported with the following command: `keytool -import -trustcacerts -alias "sensible-name-for-ca" -file directory.crt -keystore $JAVA_HOME/lib/security/cacerts`

Example LDAP JAAS configuration

```
ShibUserPassAuth {
   edu.vt.middleware.ldap.jaas.LdapLoginModule required
      ldapUrl="ldap://ldap1.example.org:389 ldap://ldap2.example.org:389 ldap://ldap3.example.org:389"
      baseDn="ou=people,dc=example,dc=org"
      tls="true"
      userFilter="uid={0}"
   ;
};
```

Additional information and more advanced configuration options may be found on the Virginia Tech Middleware LDAP Utils home page. Thanks go to Virginia Tech for providing us with a high-quality LDAP Login Module implementation.

With this login module you should enable logging for the package `edu.vt.middleware.ldap`.

Kerberos Login Module Specification

Kerberos support is only known to be available in Oracle's JREs and OpenJDK through the login module `com.sun.security.auth.module.Krb5LoginModule`. Other implementations may be available in JREs/containers from other vendors. You may check the documentation for your JRE /container for more information.

The Oracle/OpenJDK Krb5LoginModule **does not** support verifying the acquired ticket against a keytab. It therefore does not protect against the KDC being spoofed. (The keyTab option to the module is used in place of a password for acquiring a ticket, not for verifying it. Using it for password authentication does not make sense.)

If the communication between the IdP and the KDC can be eavesdropped, then an attacker can forge reply packets from the KDC and trick the IdP into accepting any password. Therefore, if you are using Krb5LoginModule, make sure that the network between the IdP and the KDC is reasonably secure.

Kerberos will return a principal name of the format *userid@CAMPUS.EDU* (i.e. includes the realm). This could cause problems if you are then pulling attributes out of an LDAP directory for that principal, because there might not be an indexed attribute in the LDAP directory of that form. So the realm (the @CAMPUS.EDU) needs to be removed from the principal name to have just the userid left for lookup in the directory. Here is one example of how to do this in the attribute resolver configuration.

Example Kerberos JAAS configuration

```
ShibUserPassAuth {
   com.sun.security.auth.module.Krb5LoginModule required;
};
```

Additional information and more advanced configuration options may be found in Oracle's Krb5LoginModule documentation.

Advanced Configuration Options

Using a Different JAAS Configuration

If you are deploying more than one IdP within a single container, and each IdP needs to use a different JAAS configuration, you will need to set the name of the configuration for each IdP.

The Servlet that performs the username/password authentication can be configured to use a JAAS configuration that is different from the default `ShibUserPassAuth`. This Servlet accepts an init-parameter called `jaasConfigName`. The value of this parameter is the JAAS configuration used by the servlet to authenticate the user.

Example web.xml snippet that overrides the default JAAS configuration used to authenticate a user.

```
<servlet>
      <servlet-name>UsernamePasswordAuthHandler</servlet-name>
      <servlet-class>edu.internet2.middleware.shibboleth.idp.authn.provider.UsernamePasswordLoginServlet<
/servlet-class>
      <init-param>
         <param-name>jaasConfigName</param-name>
         <param-value>ShibUserPassAuthTest</param-value>
      </init-param>
  </servlet>
```

If you have more than one IdP in the same container, put all JAAS config elements in one login.config file and have each IdP use that file, and then vary which configuration is used by the jaasConfigName in the respective web.xml file for each IdP installation.

Using a Different Login Page

The Servlet that performs the username/password authentication can be configured to use a login that is different from the default `login.jsp`. This Servlet accepts an init-parameter called `loginPage`. The value of this parameter is the path, rooted at the sevlet context, to the login JSP page to which the servlet will redirect in order to collect the user's credentials.

Example web.xml snippet that overrides the default login page used to authenticate a user.

```
<servlet>
      <servlet-name>UsernamePasswordAuthHandler</servlet-name>
      <servlet-class>edu.internet2.middleware.shibboleth.idp.authn.provider.UsernamePasswordLoginServlet<
/servlet-class>
      <init-param>
         <param-name>loginPage</param-name>
         <param-value>specialPages/login.jsp</param-value>
      </init-param>
  </servlet>
```

Stacking Login Modules

Sometimes it is desirable to authenticate users by either using different mechanisms (e.g. try LDAP then Kerberos), or by the same mechanism but with different parameters (e.g. different LDAP search bases). This may be accomplished by stacking 2 or more JAAS modules and configuring the Flag on each to allow for the desired fall-through behavior. Legal Flag values are: Required; Requisite; Sufficient; and Optional. See JAAS Configuration JavaDoc for details.

Here is an example where users are authenticated by checking 2 different LDAP search bases (the assumption here being that a full subtree search from the directory root is undesirable):

Example JAAS config for stacked LDAP login modules with different search bases

```
ShibUserPassAuth {

   edu.vt.middleware.ldap.jaas.LdapLoginModule sufficient
      ldapUrl="ldap://ldap.example.org:389"
      baseDn="ou=Employees,dc=example,dc=org"
      ssl="true"
      userFilter="uid={0}";

   edu.vt.middleware.ldap.jaas.LdapLoginModule sufficient
      ldapUrl="ldap://ldap.example.org:389"
      baseDn="ou=Students,dc=example,dc=org"
      ssl="true"
      userFilter="uid={0}";

};
```

The following additional example (provided by Mathias Rufer from Université de Neuchâtel) shows the configuration that would be needed in case the user's credentials shall be checked against Kerberos first and then against Microsoft Active Directory (AD).
The goals of this approach are:

1. Users shall only be able to authenticate with their Pre Wind2k username (without @domain)
2. Username shall be case insensitive
3. User must belong to a specific OU subtree in LDAP for authentication
4. Account shall be disabled after X failed login attempts

These goals require Kerberos **and** AD/LDAP be used for authentication. For this to work all KDCs must explicitly be specified in /etc/krb5.conf because dns_lookup_kdc=true won't work with JAAS.

Example JAAS config for stacked Keberos + AD/LDAP login modules

```
ShibUserPassAuth {

// See: https://spaces.internet2.edu/display/SHIB2/IdPAuthUserPass

//kerberos authentication to assure username/password are valid
//ensure that native kerberos is configured
   com.sun.security.auth.module.Krb5LoginModule requisite;

// LDAP authentication to assure user belongs to OU XYZ
   edu.vt.middleware.ldap.jaas.LdapLoginModule required
      ldapUrl="ldap://ldap.example.org:389"
      ssl="false"
      tls="false"
      baseDn="ou=users,dc=example,dc=org"
      subtreeSearch="true"
      userFilter="sAMAccountName={0}"
      bindDn="<ldapservicedn>"
      bindCredential="<password>";

};
```

There are many possible advanced configurations available via module stacking. Some additional JAAS configuration resources can be found here:

JAAS Configuration JavaDoc documentation
JAAS Reference Guide Examples

Failover configuration

If attribute resolving is configured to use multiple LDAP servers for failover reasons, the same should be done for authentication.

Example JAAS config for two LDAP servers used for failover

```
ShibUserPassAuth {
  edu.vt.middleware.ldap.jaas.LdapLoginModule required
     ldapUrl="ldap://ldap1.example.org:636 ldap://ldap2.example.org:636"
     ssl="true"
     connectionHandler="edu.vt.middleware.ldap.handler.DefaultConnectionHandler
{{connectionStrategy=ACTIVE_PASSIVE}{connectionRetryExceptions=javax.naming.CommunicationException}}"
     timeout="1000"
     baseDn="ou=users,dc=example,dc=org"
     subtreeSearch="true"
     userFilter="uid={0}"
     bindDn="<ldapservicedn>"
     bindCredential="<password>";
};
```

The timeout property specifies the time in milliseconds that connection operations will block. If you prefer to use an active-active configuration you can set connectionStrategy=RANDOM to select a random host for each authentication. Failover will still occur if any of the hosts are unavailable. If you are using TLS then you will need to specify a different connectionHandler:

```
     connectionHandler="edu.vt.middleware.ldap.handler.TlsConnectionHandler{{connectionStrategy=ACTIVE_PASSIVE}
{connectionRetryExceptions=javax.naming.CommunicationException}}"
```

The test to make sure this works is to have all LDAP servers running, start up the IdP and then disconnect the first LDAP server. If authentication and attribute resolution still works without an unreasonable long delay.

Example config using failover and certificates issued by private CA

```
ShibUserPassAuth {

// UA AD Auth
   edu.vt.middleware.ldap.jaas.LdapLoginModule sufficient
      ldapUrl="ldap://adua01.ua.edu:3268 ldap://adua032.ua.edu:3268"
      baseDn="dc=ad,dc=ua,dc=edu"
      bindDn="cn=uashib,ou=service,dc=ad,dc=ua,dc=edu"
      bindCredential="••••••••••"
      subtreeSearch="true"
// Directly reference imported certificate for CA used to create/sign server certs
      sslSocketFactory="{trustCertificates=file:/opt/shibboleth-idp/trustedservercerts/UA_AD_CA.pem}"
      ssl="false"
      tls="true"
      userField="sAMAccountName,uaIdentifier";

};
```

Handle User account status when using OpenLDAP Password Policy overlay

The OpenLDAP Password Policy overlay controls passwords' expiration. To be able to inform users whether a login is failed because of invalid credentials or an expired password, the ldaptive ldap JAAS library can be used for authentication.

Add the ldaptive jar to the unpacked IdP distribution's `lib` folder.

Modify config.login to use correct `authenticationControls` and `authenticationResponseHandlers`. ldaptive allows many other parameters, including failover, SASL authentication and so forth.

```
ShibUserPassAuth {
    org.ldaptive.jaas.LdapLoginModule sufficient
    ldapUrl="ldap://ldap1.test.com:389"
    baseDn="ou=people,dc=example,dc=com"
    bindDn="cn=idp,ou=agents,dc=example,dc=com"
    subtreeSearch="true"
    bindCredential="secret"
    useStartTLS="true"
    authenticationControls="org.ldaptive.control.PasswordPolicyControl"
    authenticationResponseHandlers="org.ldaptive.auth.ext.PasswordPolicyAuthenticationResponseHandler"
    userFilter="(uid={user})";
}
```

Modify `login.jsp` to display a meaningful message (following IdPAuthUserPassLoginPage):

```
<%@ page import="edu.internet2.middleware.shibboleth.idp.authn.LoginHandler" %>

<% if (request.getAttribute(LoginHandler.AUTHENTICATION_EXCEPTION_KEY) != null) {
        Throwable ex = (Throwable)request.getAttribute(LoginHandler.AUTHENTICATION_EXCEPTION_KEY);
        if (ex.getMessage().contains("PASSWORD_EXPIRED")){ %>
      Password expired</p>
    <p>
      Password can be changed at:
      <a href="https://my_change_password_url/login.php">change_password</a>.

      <% }else{ %>
      Authentication failed
    <% } %>
```

Re-deploy the IdP with the `install.sh` or `install.bat` script.

To double check the text message send by ldaptive to the login page, in order to tailor the `contains` method, enable debug of ldaptive in `logging.xml`:

```
<logger name="org.ldaptive" level="INFO"/>
```

Internationalization for username/password

If you like to use accented characters (or other iso-latin 1 symbols) in your username/password,
you need to have the following line in login.jsp:

```
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1" >
```

A different solution to support all UTF-8 chars in username/password: Extend the file src/main/webapp/WEB-INF/web.xml.

Add this filter to web.xml

```
<filter>
<filter-name>encoding-filter</filter-name>
<filter-class>
org.springframework.web.filter.CharacterEncodingFilter
</filter-class>
<init-param>
<param-name>encoding</param-name>
<param-value>UTF-8</param-value>
</init-param>
</filter>

<filter-mapping>
<filter-name>encoding-filter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

Finally, reinstall the IdP using the install.sh or install.bat script.