2.4. Writing a custom authenticator

2.3. Start TLS with ApacheDS | 2. Authentication & Authorization | 2.5. Authorization

Work in progress

This site is in the process of being reviewed and updated.

Warning

This page is out of date

Using custom authenticators

Authenticator SPI provides a way to implement your own authentication mechanism, for instance simple mechanism using password encryption such as MD5 or SHA1, or SASL mechanism. See the following example:

```
import javax.naming.NamingException;

import org.apache.directory.server.core.authn.AbstractAuthenticator;
import org.apache.directory.server.core.authn.LdapPrincipal;
import org.apache.directory.server.core.jndi.ServerContext;
import org.apache.directory.shared.ldap.aci.AuthenticationLevel;
import org.apache.directory.shared.ldap.name.LdapDN;

public class CustomAuthenticator extends AbstractAuthenticator {
        public CustomAuthenticator() {
                // create authenticator that will handle "simple" authentication mechanism
                super("simple");
        }

        public void init() throws NamingException {
                // ...
        }

        public LdapPrincipal authenticate(LdapDN bindDn, ServerContext ctx) throws NamingException {
                // ...

                LdapPrincipal principal = AbstractAuthenticator.createLdapPrincipal(bindDn.toNormName(),
AuthenticationLevel.SIMPLE);
                // ..
                return principal;

        }
}
```

The authenticator class has to extend the org.apache.directory.server.core.authn.AbstractAuthenticator. This class needs to have a no-argument constructor that calls the super() constructor with parameter the authentication mechanism it is going to handle. In the above example, MyAuthenticator class is going to handle the simple authentication mechanism.

You can optionally implement the init() method to initialize your authenticator class. This will be called when the authenticator is loaded by ApacheDS during start-up.

When a client performs an authentication, ApacheDS will call the authenticate() method. You can get the client authentication info from the server context. After you authenticate the client, you need to return the authorization id. If the authentication fails, you should throw an LdapNoPermissionException.

When there are multiple authenticators registered with the same authentication type, ApacheDS will try to use them in the order it was registered. If one fails it will use the next one, until it finds one that successfully authenticates the client.

To tell ApacheDS to load your custom authenticators, you need to specify it in the server.xml. You can also optionally specify the location of a .properties file containing the initialization parameters. See the following example:

EXAMPLE BELOW IS NO LONGER VALID WITH XML CONFIGURATION

```
server.authenticators=myauthenticator yourauthenticator

server.authenticator.class.myauthenticator=com.mycompany.MyAuthenticator
server.authenticator.properties.myauthenticator=myauthenticator.properties

server.authenticator.class.yourauthenticator=com.yourcompany.YourAuthenticator
server.authenticator.properties.yourauthenticator=yourauthenticator.properties
```