IdPEnableECP

The ECP profile is a SOAP-based interaction with the IdP that supports non-browser application uses of SAML.

The issues associated with ECP go beyond just basic configuration, but simple enablement of the feature at the IdP is fairly simple. The V2.3 IdP includes support for ECP "out of the box" but because of how authentication works in the V2 IdP, the standard authentication setup in the IdP does not apply to the use of ECP. This will be corrected in V3, but in the meantime, the ECP profile handler requires the use of container- or web server-based authentication such that REMOTE_USER is set.

If your IdP uses the RemoteUser LoginHandler with Basic Authentication (not terribly likely, but possible), then you can extend the protection of your authentication setup to include the path to the ECP handler, which is **/idp/profile/SAML2/SOAP/ECP**

If not, then you will have to add additional configuration to your web server, Java container, etc. to protect this path. The most common mechanism for this will be HTTP Basic Authentication, and most ECP clients would typically support that. Using client certificates is certainly a possibility as well, but you would likely need control over the client to ensure support for that.

We can't really document how you would set up authentication because it is specific to your web server and your authentication source(s). One example would be JAAS. The IdP uses JAAS login modules in its UsernamePassword handler to accomplish authentication, and most Java containers can also be configured to use the same JAAS configuration.

Assuming that's possible, you can modify the IdP's deployment descriptor to enable container-managed authentication for the ECP endpoint. The best way to do this is to copy **web.xml** from **src/main/webapp/WEB-INF/web.xml** into **/opt/shibboleth-idp/conf/web.xml** (or wherever your installation lives) and then modify it to include something like this:

Excerpt from an extended web.xml

```
    <security-constraint>
        <display-name>Shibboleth IdP</display-name>
        <web-resource-collection>
            <web-resource-name>ECP</web-resource-name>
            <url-pattern>/profile/SAML2/SOAP/ECP</url-pattern>
            <http-method>GET</http-method>
            <http-method>POST</http-method>
        </web-resource-collection>
        <auth-constraint>
            <role-name>*</role-name>
        </auth-constraint>
        <user-data-constraint>
            <transport-guarantee>CONFIDENTIAL</transport-guarantee>
        </user-data-constraint>
    </security-constraint>

    <login-config>
        <auth-method>BASIC</auth-method>
        <realm-name>ShibUserPassAuth</realm-name>
    </login-config>

  <!-- Depending on the version of tomcat in use, you may also need this - a list of security roles referenced
by this web application -->

  <security-role>
      <description>The role that is required to access the ECP area</description>
      <role-name>*</role-name>
  </security-role>
```

This is purely an example. The specifics will vary by container and your JAAS configuration. Lacking JAAS as a back-end, none of this likely applies to you.

If deploying into Tomcat, you will also need to add a definition of the JAAS Realm. According to the Tomcat documentation, this can be inserted into either an Engine, Host, or Context configuration. The easiest might be to modify the Context definition for the IdP web application itself in idp.xml: insert the following snippet:

```
      <Realm className="org.apache.catalina.realm.JAASRealm"
             appName="ShibUserPassAuth"
             userClassNames="edu.vt.middleware.ldap.jaas.LdapPrincipal"
             roleClassNames="edu.vt.middleware.ldap.jaas.LdapRole"/>
```

- Note: the `appName` attribute value must match the declaration name in login.config
- Note: this snippet uses the user and role class names as per the VtLdapJAAS documentation.

The final step would also be to set the `java.security.auth.login.config` Java property to point to your `login.config` file. This could be either done in the Tomcat startup configuration, or, alternatively, if you are using the `UsernamePassword` login handler, the initialization code for this login handler would set the `java.security.auth.login.config` Java property to the value of the `jaasConfigurationLocation` parameter passed to this login handler, so no further action is required.