IdPJetty9Prepare

Preparing Jetty 9 for the Shibboleth Identity Provider

Within this document the macro `IDP_SRC` will be used to refer to the location of the expand IdP distribution directory. The macro `IDP_HOME` will be used to refer to IdP installation directory (as given during the installation process). The macro `JETTY_HOME` will be used to refer to the location of the Jetty installation directory.

TODO: need to revise these docs to reflect use of JETTY_BASE

Version Requirements

- Jetty 9.2.1+
- Java 6 or later

Required Configuration

- Jetty listens on ports 8080 and 8443 for user-facing web traffic by default. You will most likely need to modify these ports to 80 and 443 in the **jetty.xml** and **jetty-ssl.xml** config files, and make arrangements for Jetty to run as root, or utilize the setuid extension to support the privileged ports.
- Add the following Java options to your `start.ini`(all ### is the amount of memory in megabytes to allow for the option):
  - **-XX:+UseG1GC** - this enables a garbage collector that reduces the memory requirements needed for larger metadata files
  - **-Xmx1500m** - this is the maximum amount of memory that Jetty may use, at least 1.5G is recommended for larger (>25M) metadata files
  - **-XX:MaxPermSize=128m** - (Oracle Java 7 specific option) the maximum amount of memory allowed for the permanent generation object space
- Uncomment **--exec**
- Uncomment **etc/jetty-ssl.xml** at the bottom of start.ini
- Make sure at least the following modules are enabled in start.ini: plus, servlets, annotations, jstl

Recommended Configuration

- Jetty will use **/tmp** as a staging area for unpacking the warfile, and if you have cron jobs sweeping that for old files, your IdP can be disrupted. You will probably want to create a **tmp** directory under Jetty itself, and set **-Djava.io.tmpdir=tmp** in your **start.ini**
- The Jetty distribution ships with a number of example applications located in the *JETTY_HOME/webapps* directory and deployment descriptors located in *JETTY_HOME/contexts*. You should remove all of these unless you are specifically using them.

Supporting SOAP Endpoints

Most new deployments without legacy needs will not need to support back-channel SOAP communication. The most common case requiring this feature is support for legacy Shibboleth SPs using SAML 1.1 that perform attribute queries using SOAP.

If you do need this support, these connections require special security properties which are not appropriate for user-facing/browser use. Therefore an additional endpoint must be configured.

1. Copy the jetty9-dta-ssl-1.0.0.jar (asc) to *JETTY_HOME/lib/ext*.
2. Create the file *JETTY_HOME/etc/jetty-shibboleth.xml* and place something like the following content in it:

```
<Configure id="Server" class="org.eclipse.jetty.server.Server">

  <!-- =========================================================== -->
  <!-- Configure a TLS (SSL) Context Factory                       -->
  <!-- This configuration must be used in conjunction with jetty.xml -->
  <!-- and either jetty-https.xml or jetty-spdy.xml (but not both)   -->
  <!-- =========================================================== -->
  <New id="shibContextFactory" class="net.shibboleth.utilities.jetty9.
DelegateToApplicationSslContextFactory">
    <Set name="KeyStorePath">IDP_HOME/credentials/idp.jks</Set>
    <Set name="KeyStorePassword">PASSWORD</Set>
    <Set name="EndpointIdentificationAlgorithm"></Set>
    <Set name="excludeProtocols">
      <Array type="String">
         <Item>SSLv3</Item>
      </Array>
    </Set>
  </New>

  <!-- =========================================================== -->
  <!-- Create a TLS specific HttpConfiguration based on the        -->
  <!-- common HttpConfiguration defined in jetty.xml               -->
  <!-- Add a SecureRequestCustomizer to extract certificate and    -->
  <!-- session information                                         -->
  <!-- =========================================================== -->
  <New id="shibHttpConfig" class="org.eclipse.jetty.server.HttpConfiguration">
    <Arg><Ref refid="httpConfig"/></Arg>
    <Call name="addCustomizer">
      <Arg><New class="org.eclipse.jetty.server.SecureRequestCustomizer"/></Arg>
    </Call>
  </New>

  <!-- =========================================================== -->
  <!-- Add a HTTPS Connector.                                      -->
  <!-- Configure an o.e.j.server.ServerConnector with connection   -->
  <!-- factories for TLS (aka SSL) and HTTP to provide HTTPS.      -->
  <!-- All accepted TLS connections are wired to a HTTP connection.-->
  <!--                                                             -->
  <!-- Consult the javadoc of o.e.j.server.ServerConnector,        -->
  <!-- o.e.j.server.SslConnectionFactory and                       -->
  <!-- o.e.j.server.HttpConnectionFactory for all configuration    -->
  <!-- that may be set here.                                       -->
  <!-- =========================================================== -->
  <Call id="httpsConnector" name="addConnector">
    <Arg>
      <New class="org.eclipse.jetty.server.ServerConnector">
        <Arg name="server"><Ref refid="Server" /></Arg>
          <Arg name="factories">
            <Array type="org.eclipse.jetty.server.ConnectionFactory">
              <Item>
                <New class="org.eclipse.jetty.server.SslConnectionFactory">
                  <Arg name="next">http/1.1</Arg>
                  <Arg name="sslContextFactory"><Ref refid="shibContextFactory"/></Arg>
                </New>
              </Item>
              <Item>
                <New class="org.eclipse.jetty.server.HttpConnectionFactory">
                  <Arg name="config"><Ref refid="shibHttpConfig"/></Arg>
                </New>
              </Item>
            </Array>
          </Arg>
          <Set name="host"><Property name="jetty.host" /></Set>
          <Set name="port">8443</Set>
          <Set name="idleTimeout"><Property name="https.timeout" default="30000"/></Set>
          <Set name="soLingerTime"><Property name="https.soLingerTime" default="-1"/></Set>
        </New>
    </Arg>
  </Call>
</Configure>
```

3. Replace `IDP_HOME` with the IdP home directory entered during installation.
4. Replace `PASSWORD` with the password for the IdP key entered during installation.
5. Add `etc/jetty-shibboleth.xml` to your Jetty `start.ini` file (towards the bottom of the file you should see other configuration files listed)

Deploying the IdP

In order to deploy the IdP Jetty must be informed of the location of the IdP war. This can be done by:

1. Create the file *JETTY_HOME/webapp/idp.xml* and place the following content in it (replace `IDP_HOME` with your IdP's home directory):

```
<Configure class="org.eclipse.jetty.webapp.WebAppContext">
  <Set name="war">IDP_HOME/war/idp.war</Set>
  <Set name="contextPath">/idp</Set>
  <Set name="extractWAR">false</Set>
  <Set name="copyWebDir">false</Set>
  <Set name="copyWebInf">true</Set>
</Configure>
```