Jetty94

Using Jetty 9.4

These pages are examples and do not reflect any normative requirements or assumptions on the part of the IdP software and may be a mix of suggestions from both the project team and deployers. You should take any of this advice with a grain of local salt and consider general security /deployment considerations appropriate to the use of web software in your local environment.

The official information about containers and versions we support is solely maintained on the SystemRequirements page. If you wish to operate without complete responsibility for your Java servlet container, you may consider the Windows package we provide that includes an embedded container.

The following conventions are used this document:

- /opt/shibboleth-idp is used to indicate that an absolute path to the IdP installation directory is required
- `idp.home` refers to the IdP installation directory (as specified during the installation process)
- `JETTY_HOME` refers to the location of the Jetty installation (jetty-dist-$VERSION)
- `JETTY_BASE` refers to the directory containing your deployment-specific Jetty configuration files
- All paths are relative to `JETTY_BASE` unless otherwise noted

We strongly recommend placing all IdP-specific Jetty configuration under `JETTY_BASE` to facilitate Jetty upgrades. Do **not** place that directory inside the IdP installation directory; they can be siblings as desired.

The examples on this page are based on the use of a number of files that are not included in the Jetty distribution but are part of an artifact we have made available in our Nexus repository. The examples won't work as is without starting from that complete set of example files.

We may publish it in a more "official" capacity in the future, but for now it's simply an example to build on. It includes some custom Jetty "modules" that help support a simpler configuration and come packaged with logging libraries and other pieces that make the example here much simpler to explain.

Version Notes

The latest stable version of Jetty should be used.

Java 8 is required for this version. Java 11 is probably more advisable, but do review LDAPonJava>8

This release includes a number of configuration changes from earlier releases, mostly refactoring and property renaming. Starting from scratch is advisable if upgrading from 9.2 or 9.3.

Required Configuration

Configure Jetty Modules and JVM Settings

The bulk of the configuration is established by setting properties in "ini" files that are combined in the start.d directory. Some of the properties are defined by Jetty and configure built-in modules and others are specific to the IdP and configure the custom modules we created.

File(s): start.d/*start.ini*

This is a file you can create to add any specific options you want to use in addition to the defaults, such as enabling additional modules or setting JVM or other system properties.

start.ini

```
# Any other required Jetty modules...

# Allows setting Java system properties (-Dname=value)
# and JVM flags (-X, -XX) in this file
# NOTE: spawns child Java process
--exec

# Uncomment if IdP is installed somewhere other than /opt/shibboleth-idp
#-Didp.home=/path/to/shibboleth-idp

# Newer garbage collector that reduces memory needed for larger metadata files
-XX:+UseG1GC

# Maximum amount of memory that Jetty may use, at least 1.5G is recommended
# for handling larger (> 25M) metadata files but you will need to test on
# your particular metadata configuration
-Xmx1500m

# Prevent blocking for entropy.
-Djava.security.egd=file:/dev/urandom

# Set Java tmp location
-Djava.io.tmpdir=tmp
```

Configure HTTP/HTTPS Connectors

File(s): *credentials/idp-userfacing.p12*, *start.d/idp.ini*

The basic HTTP/HTTPS port, address, etc. configuration is handled within the custom "idp" module and the idp.ini property file.

The example below shows some of the basic properties you can use to configure networking and TLS credentials.

One challenge remains that if you want to use standard ports, you would need to pick one of these options to avoid running as root:

1. Use the setuid extension to support listening on the privileged ports as a non-root user.
2. Use a port forwarding approach (load balancer, iptables rules, etc).

idp.ini

```
# ---------------------------------------
# Module: idp
# Shibboleth IdP
# ---------------------------------------
--module=idp

## Keystore file path (relative to $jetty.base)
jetty.sslContext.keyStorePath=../credentials/idp-userfacing.p12
## Truststore file path (relative to $jetty.base)
jetty.sslContext.trustStorePath=../credentials/idp-userfacing.p12

## Keystore type
jetty.sslContext.keyStoreType=PKCS12
## Truststore type and provider
jetty.sslContext.trustStoreType=PKCS12

## Keystore password
jetty.sslContext.keyStorePassword=changeit
## Truststore password
jetty.sslContext.trustStorePassword=changeit
## KeyManager password
jetty.sslContext.keyManagerPassword=changeit

## Deny SSL renegotiation
jetty.sslContext.renegotiationAllowed=false

## Connector host/address to bind to
# jetty.ssl.host=0.0.0.0

## Connector port to listen on
jetty.ssl.port=443

# Allows use of default IdP command line tools.
jetty.http.host=127.0.0.1
jetty.http.port=80
```

The TLS credential example relies on a PKCS12 file containing the X.509 certificate and private key used to secure the HTTPS channel that users access during authentication and other browser-based message exchanges involving the IdP. This is generally the one you get from a browser-compatible CA, and the example shows it being loaded from a directory inside the `JETTY_BASE` tree.

A variety of other networking properties can be set based on the built-in Jetty http and https modules; refer to their documentation.

Configure IdP Context Descriptor

File(s): *webapps/idp.xml*

In order to deploy the IdP, Jetty must be informed of the location of the IdP war file. This file is called a context descriptor and the recommended content is provided below.

Note this file assumes the location of the IdP installation is explicitly set in the file, and controls the context path to which the application is deployed, which is **/idp** in the following configuration block.

idp.xml

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

Recommended Configuration

Jetty Logging

File(s): *start.d/idp-logging.ini, resources/logback.xml, resources/logback-access.xml*

The recommended approach is to use logback for all Jetty logging. The logback and slf4j libraries are needed to support this configuration and are included in the example artifact we publish.

If you **don't** want to use this feature, just remove the *etc/idp-logging.ini* file.

Configure logging policy for Jetty internals logging and request/access logging. Sample logback configuration files are provided for convenience and are in the resources directory.

Disable Directory Indexing*

Jetty has vulnerabilities related to directory indexing (sigh) so we suggest disabling that feature at this point. There are a few different ways this can be done (see https://webtide.com/indexing-listing-vulnerability-in-jetty/), but one method that's fairly self-contained within the IdP footprint is to modify web.xml (i.e. copy the original version from idp.home/dist/webapp/WEB-INF/web.xml to idp.home/edit-webapp/WEB-INF/web.xml) and then rebuild the war file.

web.xml addition

```
<servlet>
  <servlet-name>default</servlet-name>
  <servlet-class>org.eclipse.jetty.servlet.DefaultServlet</servlet-class>
  <init-param>
    <param-name>dirAllowed</param-name>
    <param-value>false</param-value>
  </init-param>
  <load-on-startup>0</load-on-startup>
</servlet>
```

You can place it above the existing `<servlet>` elements in the file.

*This bug has been fixed in later versions https://www.eclipse.org/lists/jetty-announce/msg00130.html

Optional Configuration

Supporting SOAP Endpoints

File(s): */opt/shibboleth-idp/credentials/idp-backchannel.p12*, *etc/idp-backchannel.xml, modules/idp-backchannel.mod, start.d/idp-backchannel.ini*

The use of the back-channel is discussed in the SecurityAndNetworking topic, and you should review that to understand whether or not you need to support this feature.

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

1. The jetty9-dta-ssl-1.0.0.jar (asc) plugin is already included in `JETTY_BASE/lib/ext`
2. We provide a backchannel module to control the feature and turn it on or off.
3. Adjust *JETTY_BASE/start.d/idp-backchannel.ini* as required:

```
# ---------------------------------------
# Module: idp-backchannel
# Shibboleth IdP Dedicated SOAP Connector
# ---------------------------------------
--module=idp-backchannel

## Backchannel connector port to listen on
# idp.backchannel.port=8443

## Backchannel keystore file path (relative to $jetty.base)
# idp.backchannel.keyStorePath=../credentials/idp-backchannel.p12

## Backchannel keystore password
# idp.backchannel.keyStorePassword=changeit

## Backchannel keystore type
# idp.backchannel.keyStoreType=PKCS12
```

4. Modify *JETTY_BASE/etc/idp-backchannel.xml* if desired. You get more control over the TLS settings if you need them, but normally this file is just used to plug in the properties we support from the ini file.

Modules

Jetty has a ton of advanced and optional functionality available in the form of modules that can be enabled selectively. They don't function in the way Apache modules do, but they're basically packaged "example" configuration files that will get copied from JETTY_HOME into JETTY_BAASE when you need them and you get "just" the minimum files needed to support the feature but keep future upgrades simple.

To enable a module, you run the Jetty start file from within `JETTY_BASE`:

```
$ cd jetty-base
$ java -jar /opt/jetty-distribution-9.4.14.v20181114/start.jar --add-to-start=modulename
```

Supporting X-Forwarded-For Natively

If you are running the Jetty engine behind a proxy or load balancer Jetty has built-in support for forwarding the client address and other details via headers using its http-forwarded module, and after enabling it as above you can edit the resulting properties file to configure it.

You may also need to add/enable a system property via "-Dorg.eclipse.jetty.util.HostPort.STRIP_IPV6=true" to prevent brackets from being placed around an IPv6 client address if this causes problems with some SPs.

Configure Jetty to listen only on HTTP

File(s): *modules/idp.mod*, *start.d/idp.ini*

If your IdP is behind Apache, you probably only need it to listen for HTTP traffic locally. At present, this involves commenting out or removing two lines from the depend section of idp.mod: the lines containing https and ssl.

Next, add modules to be loaded in start.d/idp.ini. Alternatively, create your own start.d/http.ini to keep this configuration separate. Add the following to one of these files.

--module=http

The http-forwarded module mentioned above is required so that requests coming to the IdP aren't seen as coming from localhost. Follow the instructions above to install this module or, since its defaults are acceptable, just add the following to start.d/idp.ini or start.d/http.ini:

--module=http-forwarded

Finally, make sure the jetty.http.host and jetty.http.port are set apropriately in idp.ini, or remove them from there and add them to http.ini.

jetty.http.host=127.0.0.1

jetty.http.port=8080

Since 8080 is the default http port for Jetty, It's also okay to leave this unset.