# HOWTO: Setup a Mujina Mock SAML2 Server

**Notes:**

This HOWTO is using the Nujina fork of Mujina. Nujina adds encryption of the returned SAML2 assertion when the SP public key is included in the saml request.

Terminology:

The dev environment is the SP (Service Provider)

The mujina server is the IDP (Identity Provider, sometimes IdP)

A “certificate” is a SP or IDP public key

## 1. IDP (Mujina Server) Installation

### A: Install Maven

# wget http://mirrors.ibiblio.org/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz

# tar -xzf apache-maven-3.6.3-bin.tar.gz

# mv apache-maven-3.6.3 /usr/local

# cd /usr/local

# ln -s apache-maven-3.6.3/ maven

# vi /etc/bashrc

export MVN\_HOME=/usr/local/maven

export PATH=$MVN\_HOME/bin:$PATH

### B: Install Javac If Not Already Present

# yum install java-1.8.0-openjdk-devel

### C: Install Mujina

Install mujina server from the github Nujina fork (Nujina fork encrypts the assertion coming back to the SP).

1. Establish the server base directory. This could be /*usr/local/webapps/mujina. Will refer to this as ~mujina*

2. Clone from github:

# git clone <https://github.com/Netazoic/Nujina.git> mujina

# cd mujina

3. Copy application.eo.yml to application.yml and adust params:

# cd mujina-idp/src/main/resources/

# cp application.eo.yml application.yml

The mujina install comes with standard values for IDP cert and Private Key set as strings in an application.yml file. You can use these default values, or generate new ones. Whatever you end up using as the Certificate, copy this to a file ssodev.crt for use later when setting up the SP development machines.

Default values are shown in [parens] below:

# vi application.yml

server: port: [8080]

idp: entity\_id: [http://mock-idp]

idp: base\_url: [http://localhost:8080

idp: private\_key: [MIIE...]

idp: certificate: [MIIDEz...]

idp: passphrase: [secret]

4. Build the project

# cd ~mujina

# mvn clean install

Watch for any error messages and the report at the end of the build. Should show all 4 steps completed successfully like this:

[INFO] ------------------------------------------------------------------------

[INFO] Reactor Summary for mujina 7.4.1:

[INFO]

[INFO] mujina ............................................. SUCCESS [ 2.686 s]

[INFO] mujina-common ...................................... SUCCESS [ 9.218 s]

[INFO] mujina-sp .......................................... SUCCESS [01:14 min]

[INFO] mujina-idp ......................................... SUCCESS [01:13 min]

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

### D: Start the Mock Server

# cd ~mujina/mujina-idp

# mvn spring-boot:run

You will see a series of SpringBoot output lines that should terminate with a line like the following:

14:26:40.556 12720 [main] INFO mujina.MujinaIdpApplication - Started MujinaIdpApplication in 12.266 seconds (JVM running for 24.81)

//TODO Create a init control script

### E. Test the Server Installation

To test the server browse to the IDP URL from a remote machine e.g.,

http://ssodev/SingleSignOnService

The first time you access the server it will go through initialization and may take about 20 seconds to respond. Ultimately, it should return a json message like this:

{"timestamp":"2020-08-13T23:22:06.133+0000","status":500,"error":"Internal Server Error","message":"No SAMLRequest or SAMLResponse query path parameter, invalid SAML 2 HTTP Redirect message","path":"/SingleSignOnService"}

Although this is technically an error message, it indicates that everything is working as it should.

## 2: Service Provider (SP) Setup

### A. Generate keystore file

1. On the SP box (i.e., your development machine), copy ssodev.crt to a keystore directory, and copy ~mujina/bin/mkjks.sh into keystore/bin

2. run mkjks.sh with the basename for your cert, e.g.,

# ./bin/mkjks.sh dgpdev

Enter ‘secret’, or whatever password you want for the jks file when prompted.

This will create the following files:

dgpdev.crt (public key, text)

dgpdev.pem (private key, text)

depdev.p12 (private key, pcks12 binary)

depgdev.jks (jks file with 3 entries:

dgpdev-sp : SP private key from dgpdev.p12

dgpdev-sp-crt: SP public key from dgpdev.crt

ssodev-idp: ssodev IDP public key

### B: Set SP/IDP ID and URL Params

Set values for the following params into web.xml

* **SAML2SSOServiceProviderEntityID:** can be anything; e.g., dgp-dev
* **SAML2SSOServiceProviderAssertionConsumerURL:** the url for your dev environment. Does not need to be resolvable by the IDP –- will be resolved by your browser
* **SAML2SSOIdentityProviderEntityID**: Needs to be the value you set in application.yml idp: entity\_id
* **SAML2SSOIdentityProviderURL**: The URL that your dev environment will use to talk to the mujina server. Needs to have the same base url as the value set in application.yml idp: base\_url. Value will be:

<base\_url>/SingleSignOnService

e.g.,

http://ssodev:8080/SingleSignOnService

Example param settings:

<!-- SAML2 SSO SP Parameters -->

<context-param>

<param-name>SAML2SSOServiceProviderEntityID</param-name>

<param-value>dgp-dev</param-value>

</context-param>

<context-param>

<param-name>SAML2SSOServiceProviderAssertionConsumerURL</param-name>

<param-value>http://d.dgp/home</param-value>

</context-param>

<!-- END SAML2 SSO SP Parameters -->

<!-- SAML2 SSO IDP Parameters -->

<context-param>

<param-name>SAML2SSOIdentityProviderEntityID</param-name>

<param-value>http://ssodev</param-value>

</context-param>

<context-param>

<param-name>SAML2SSOIdentityProviderURL</param-name>

<param-value>http://ssodev:8080/SingleSignOnService</param-value>

</context-param>

<!-- END SAML2 SSO IDP Parameters -->