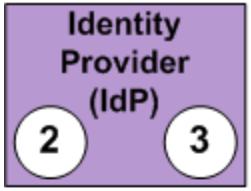
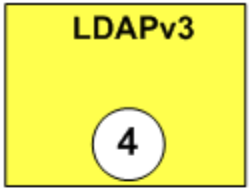
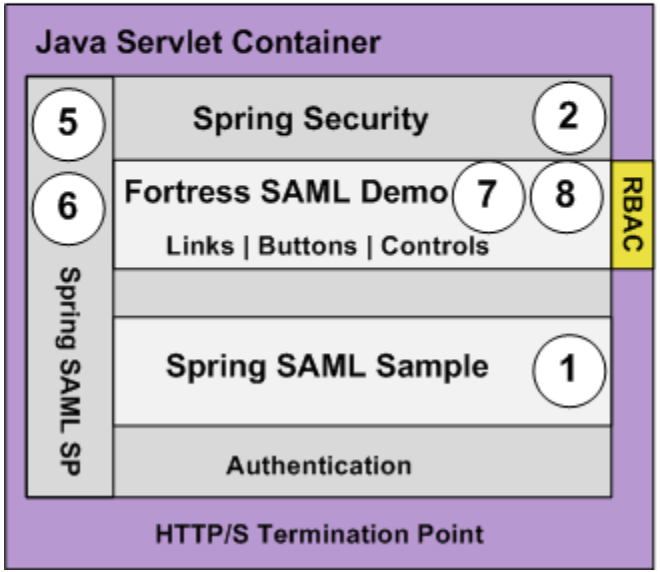
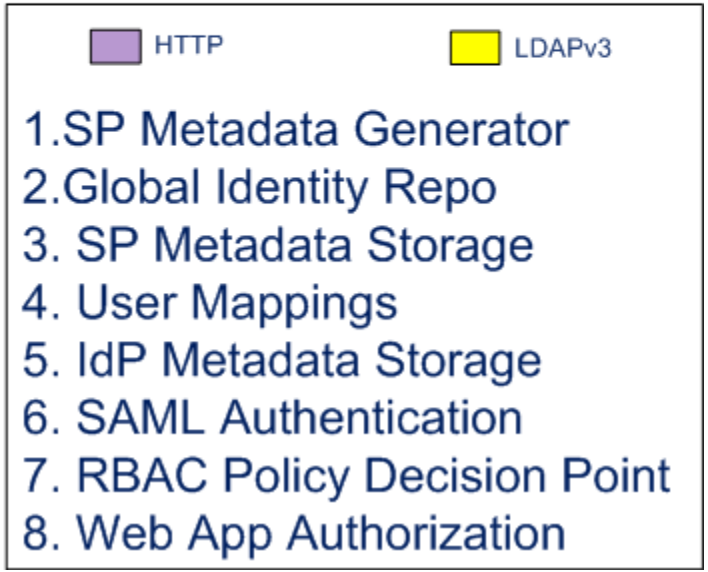
Tutorial

Apache Fortress SAML Demo









http://iamfortress.net/2015/09/01/apache-directory-fortress-saml-demo/

The Five Security Layers with SAML

1. JSSE
2. ~~Java EE Security~~ ← Turned off (for now)
3. Spring Security ← Deadbolt is now here
4. Web App Framework
5. Database Functions

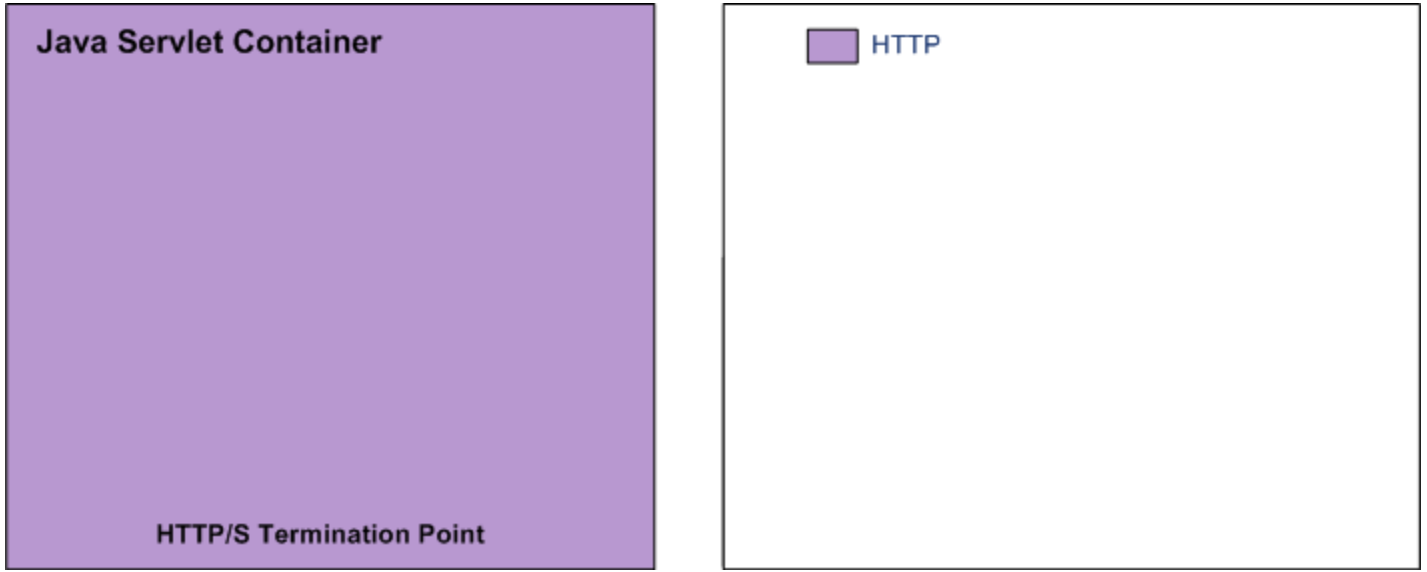
Not much to change

Two Areas of Access Control

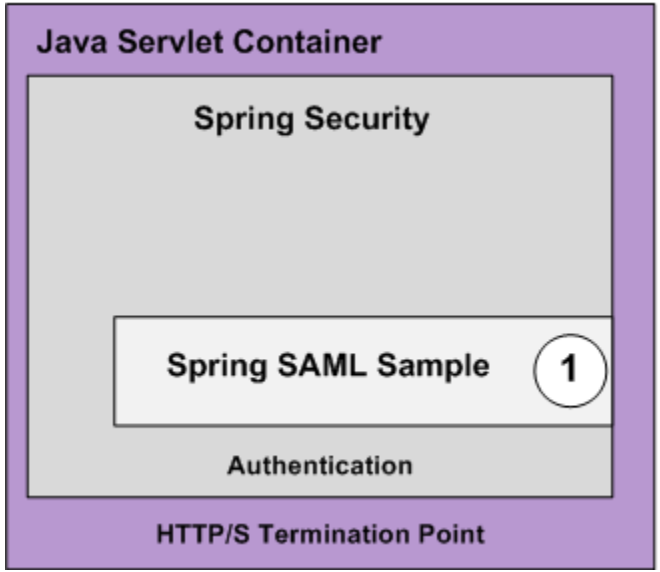
1. Spring SAML Declarative checks

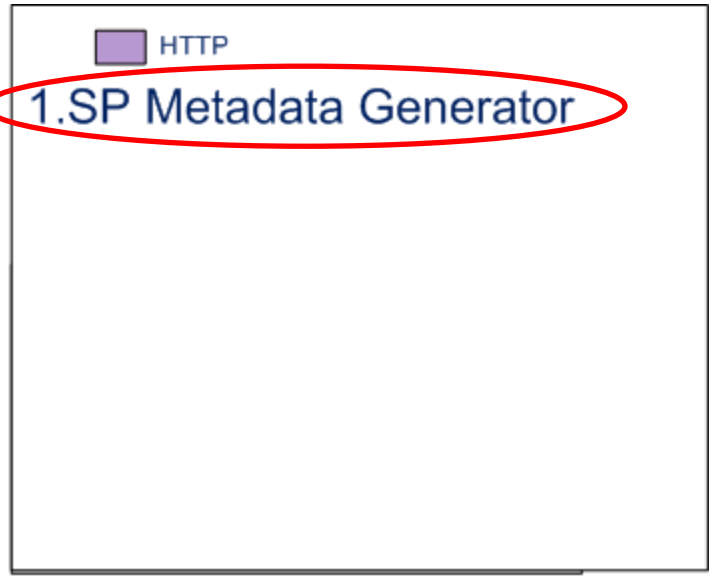
2. RBAC Permission Programmatic checks

Start with Tomcat Servlet Container



1. Deploy the Spring SAML Demo





Get the Spring SAML Demo

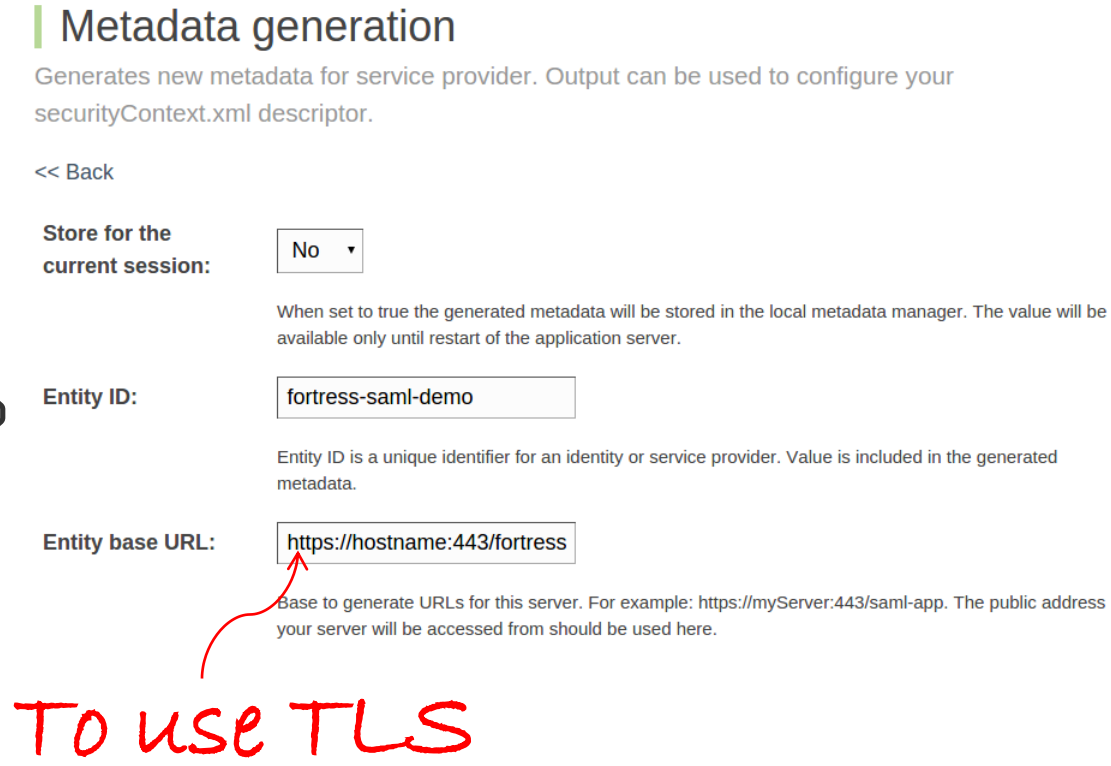
Pick one:

- spring-security-saml - Spring's SAML sample is the first place java developers should look for basic SAML 2.0 programming concepts.
- shibboleth-sample-java-sp - Unicon's sample is where ones goes to understand how to combine Spring SAML's SP with Shibboleth's IdP.

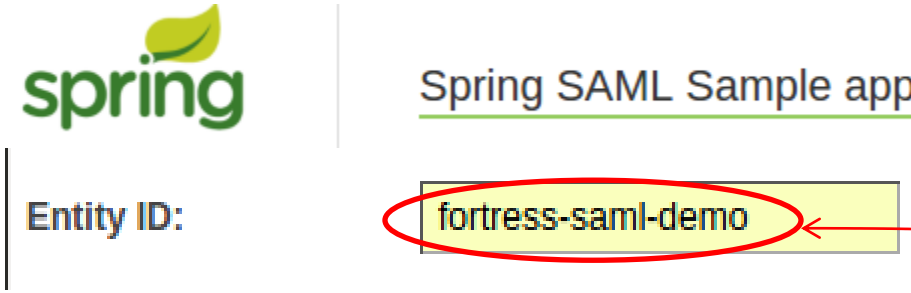
Generate SAML Service Provider Metadata

Matching Fields:

- Entity ID must match Spring config in web app
- Entity base URL must match the web app's URL.



Spring SAML Metadata Generation Tip

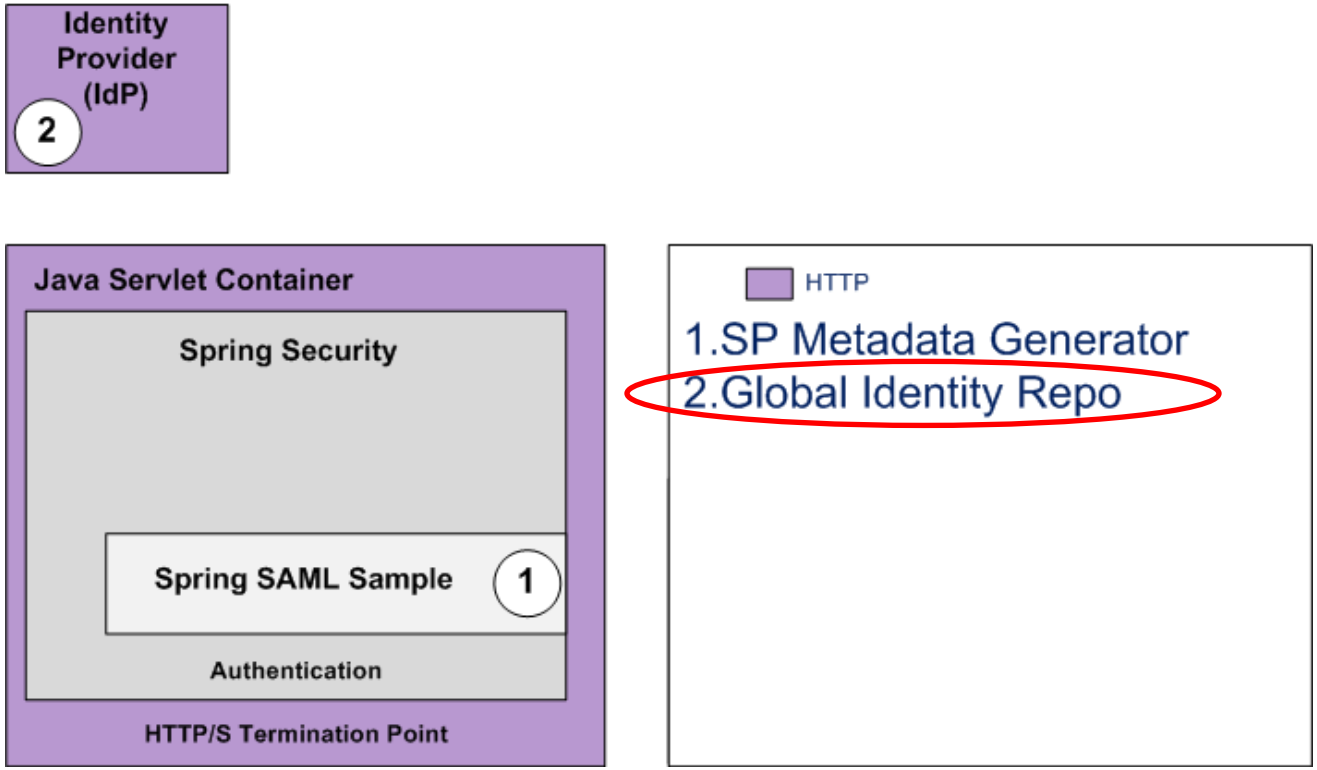


These entityId's must match

```
<bean id="metadataGeneratorFilter"
   class="org.springframework…MetadataGeneratorFilter">
  <constructor-arg>
  <bean class="org.springframework…MetadataGenerator">
<property name="entityId" value="fortress-saml-demo"/>
  </bean>
  </constructor-arg>
</bean>
```

Bind the service provider with the IdP.

2. Setup Global Identity Provider



Setup SSOCircle SAMLv2.0 IdP

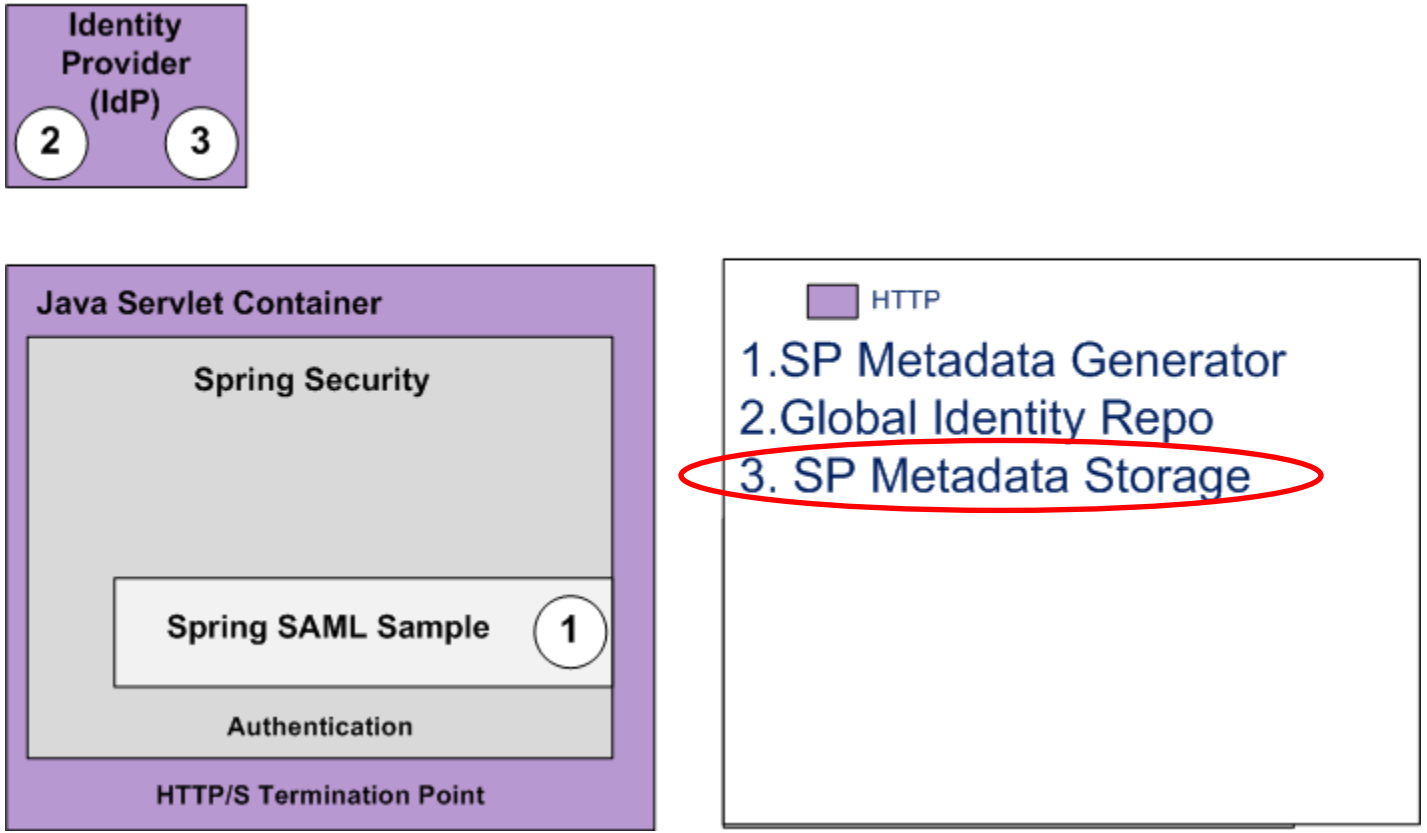
Creating your Identity with SSOCircle (from their website)

For creating your account you need to follow a few steps:

- Register at the SSOCircle SAMLv2.0 Identity Provider
- Provide the required data
- Agree to the Terms of Use
- After successful creation you will receive an email asking for confirmation of your registration. Confirm by navigating to the link supplied in the email.
- Now your account is activated and ready for use.

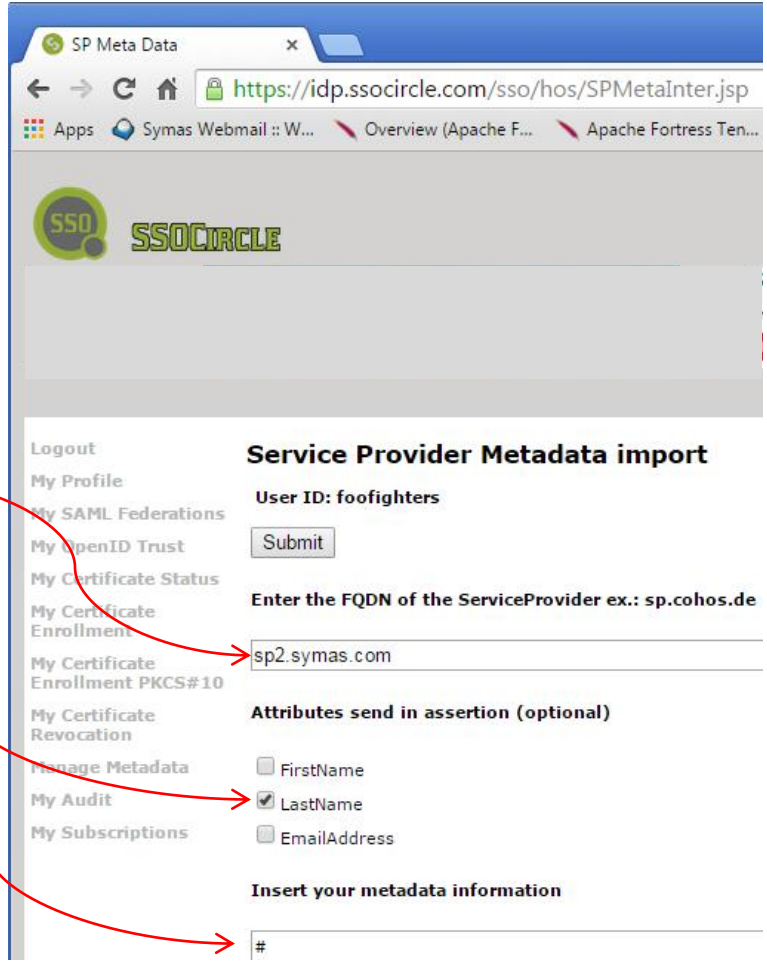
http://www.ssocircle.com/en/portfolio/publicidp/

3. Import Service Provider Metadata into IdP



Import SP Metadata

- Logon SSOCircle
- Click on ***Manage Metadata***
- ***FQDN*** must match SP's host name
- Check the ***LastName*** box
- Paste your metadata here



Import SP Metadata Tip

The FQDN matches base url from SP metadata gen step

Spring SAML app Metadata Generation page:

Spring SAML Sample application

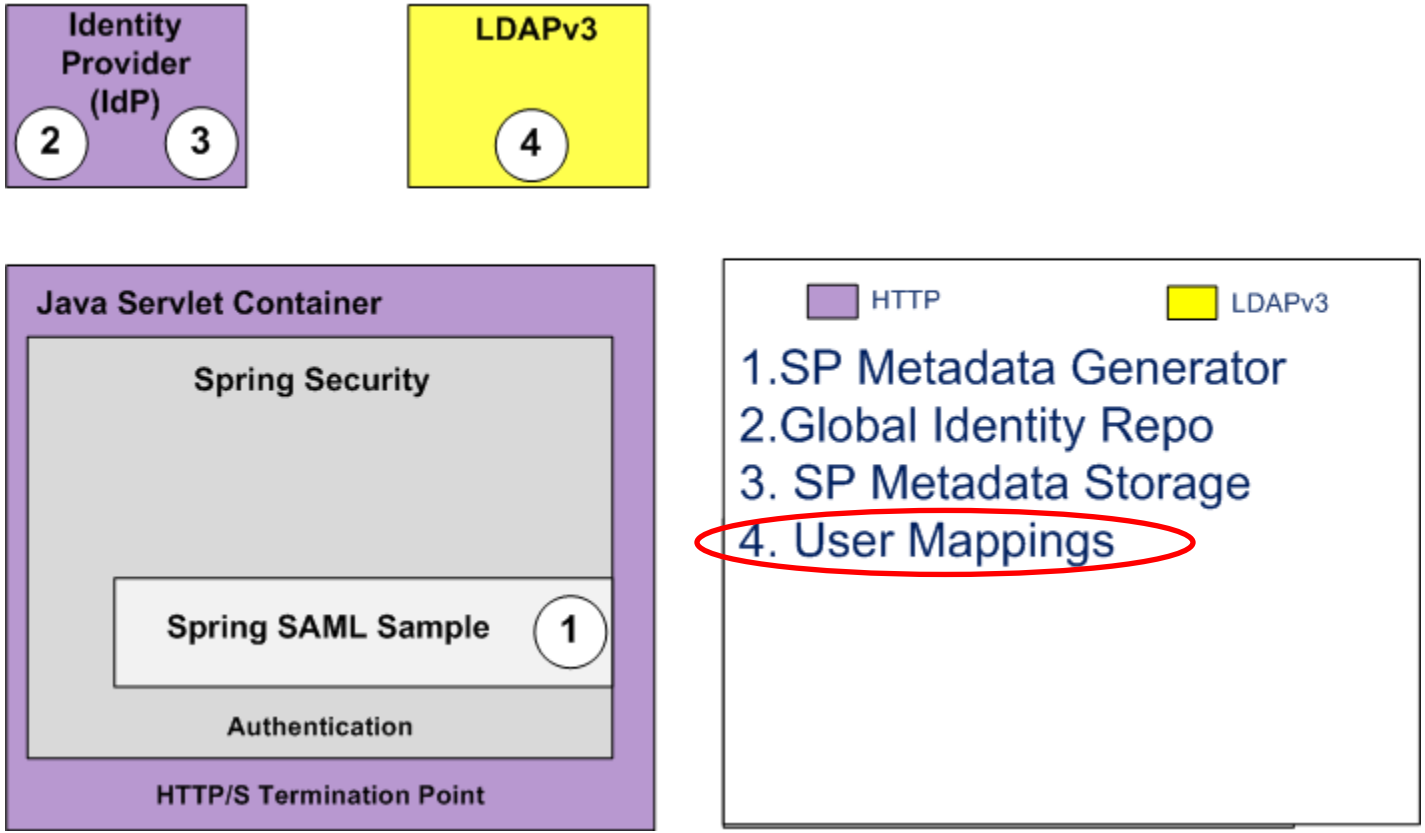
Entity base URL: http://sp2.symas.com:8080/

SSOCircle Service Provider Metadata Import page:

Enter the FQDN of the ServiceProvider ex.: sp.cohos.de

sp2.symas.com

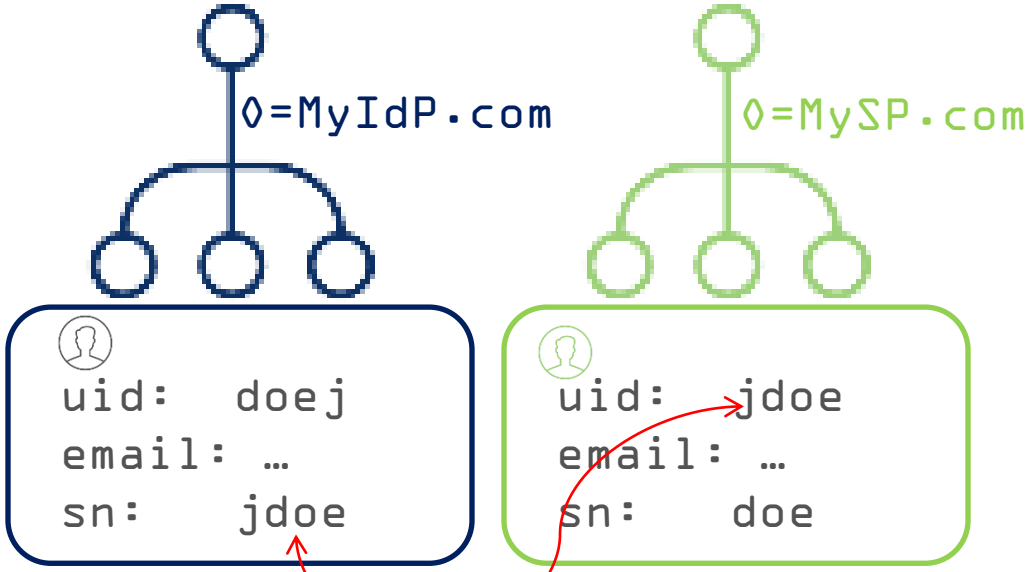
4. IdP and SP User Account Mapping



IdP and SP User Account Mapping

1. Mapping rules are specific to partners.

2. The mapping must be a one-to-one unique pairing.



fortress saml demo maps the sn on the IdP-side with uid field on the SP-side

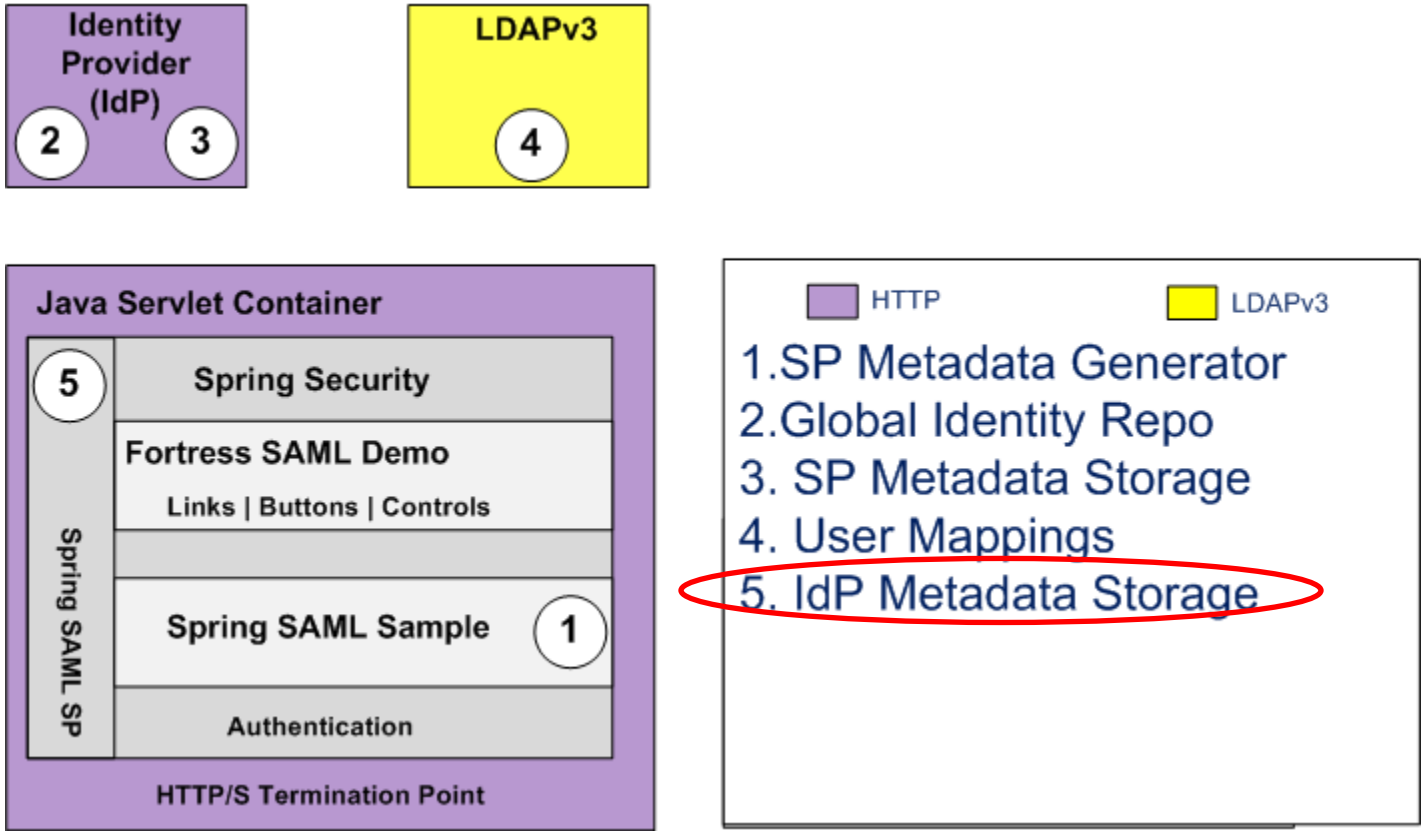
SAML Attribute Statement

host name entered during SP Metadata import

```
<?xml version="1.0" encoding="UTF-8"?><samlp:Response
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
Destination="http://sp2.symas.com:8080/fortress-saml-demo/saml/SSO"
…
<saml:AttributeStatement>
…
  <saml:Attribute Name="LastName">
  <saml:AttributeValue …
       xsi:type="xs:string">sam3</saml:AttributeValue>
  </saml:Attribute>
 </saml:AttributeStatement>
…
</samlp:Response>
```

Last Name linked to userid in rbac

5. Load IdP Metadata into Service Provider



Point SP to SAML IdP

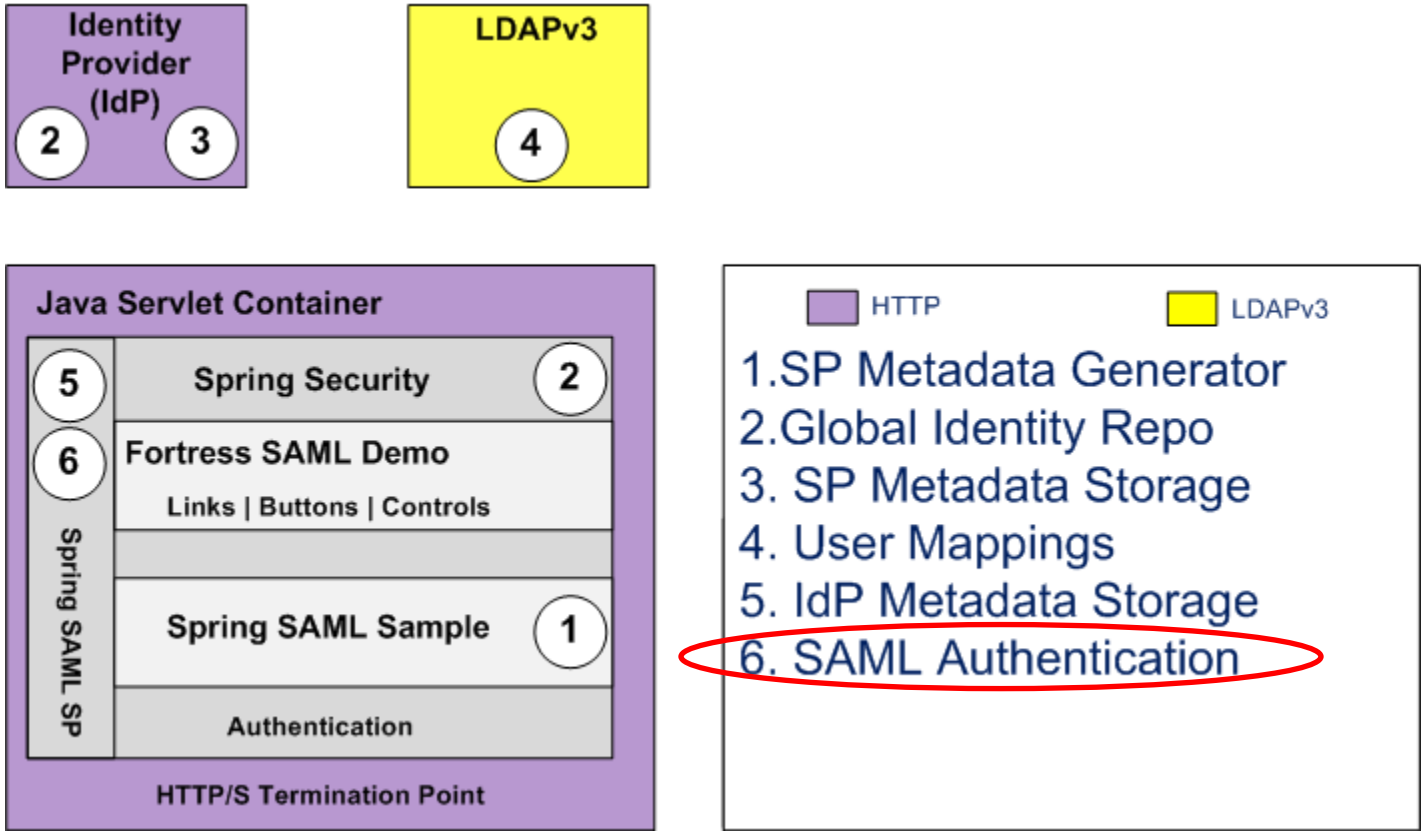
Point to the Identity Provider in securityContext.xml

```
<bean id="metadata"
    class="org.springframework.security.saml.metadata.CachingMetadataManager">
  <constructor-arg>
    <list>
      <bean class="org.opensaml.saml2.metadata.provider.HTTPMetadataProvider">
        <constructor-arg>
          <value type="java.lang.String"
```

http://idp.ssocircle.com/idp-meta.xml

```
          </value>
        </constructor-arg>
        <constructor-arg>
          <value type="int">5000</value>
          </constructor-arg>
          <property name="parserPool" ref="parserPool"/>
      </bean>
    </list>
  </constructor-arg>
</bean>
```

6. Enable Spring SAML Authentication



Enable Spring SAML Security

Add dependencies to pom:

```
<dependency>
  <groupId>org.springframework.security.extensions</groupId>
  <artifactId> spring-security-saml2-core </artifactId>
  <version>1.0.1.RELEASE</version>
  <scope>compile</scope>
</dependency>
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId> spring-security-config </artifactId>
  <version> 3.1.2.RELEASE* </version>
  <scope>compile</scope>
</dependency>
* backlog item
```

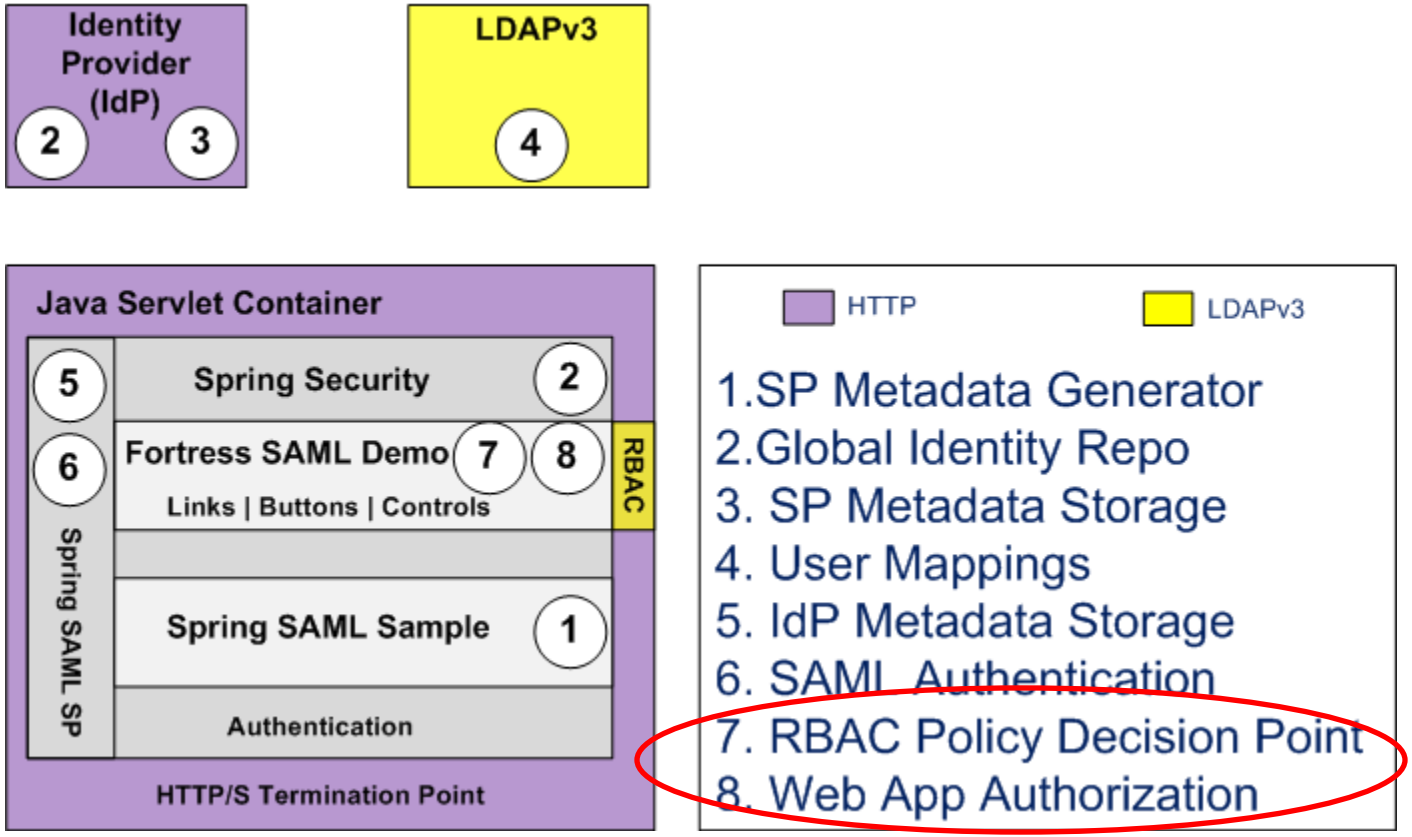
Enable SAML Authentication Filters

In the securityContext.xml

```
<security:http entry-point-ref="samlEntryPoint" use-expressions="false">
<security:intercept-url pattern="/**" access="IS_AUTHENTICATED_FULLY"/>
  <security:custom-filter before="FIRST" ref="metadataGeneratorFilter"/>
  <security:custom-filter after="BASIC_AUTH_FILTER" ref="samlFilter"/>
</security:http>

<bean id="samlFilter" class="org.springframework.security.web.FilterChainProxy">
    <security:filter-chain-map request-matcher="ant">
    <security:filter-chain pattern="/saml/login/**" filters="samlEntryPoint"/>
    <security:filter-chain pattern="/saml/logout/**" filters="samlLogoutFilter"/>
    <security:filter-chain pattern="/saml/metadata/**" filters="metadataDisplayFilter"/>
    <security:filter-chain pattern="/saml/SSO/**" filters="samlWebSSOProcessingFilter"/>
    <security:filter-chain pattern="/saml/SSOHoK/**" filters="samlWebSSOHoKProcessingFilter"/>
    <security:filter-chain pattern="/saml/SingleLogout/**" filters="samlLogoutProcessingFilter"/>
  </security:filter-chain-map>
</bean>
```

7. Setup RBAC Policy Decision Point



Enable RBAC Policy Decision Point

```
<dependency>
  <groupId>
    org.apache.directory.fortress
  </groupId>
  <artifactId>
    fortress-realm-impl
  </artifactId>
  <version>1.0</version>
</dependency>
```

Identity Propagation SAML->RBAC

1. Spring SAML filter creates security principal based on attributes found in the SAML attribute assertion.

2. Web app parses the attributes contained within principal :

```
uid=getSurName((SAMLCredential)principal.getCredentials()
);
```

3. Web app creates a new RBAC session using attribute(s) pulled from the principal :

```
j2eePolicyMgr.createSession( new User( uid ), true );
```

4. Web app pushes RBAC session into HTTP session.

isTrusted

Apache Fortress Saml Demo

- Three Pages
- Each has buttons controlled by RBAC permissions.
- One role per page.
- Users may be assigned to one or more roles.

User to Role	Page One	Page Two	Page Three
Sam*	True	True	True
Sam1	True	False	False
Sam2	False	True	False
Sam3	False	False	True

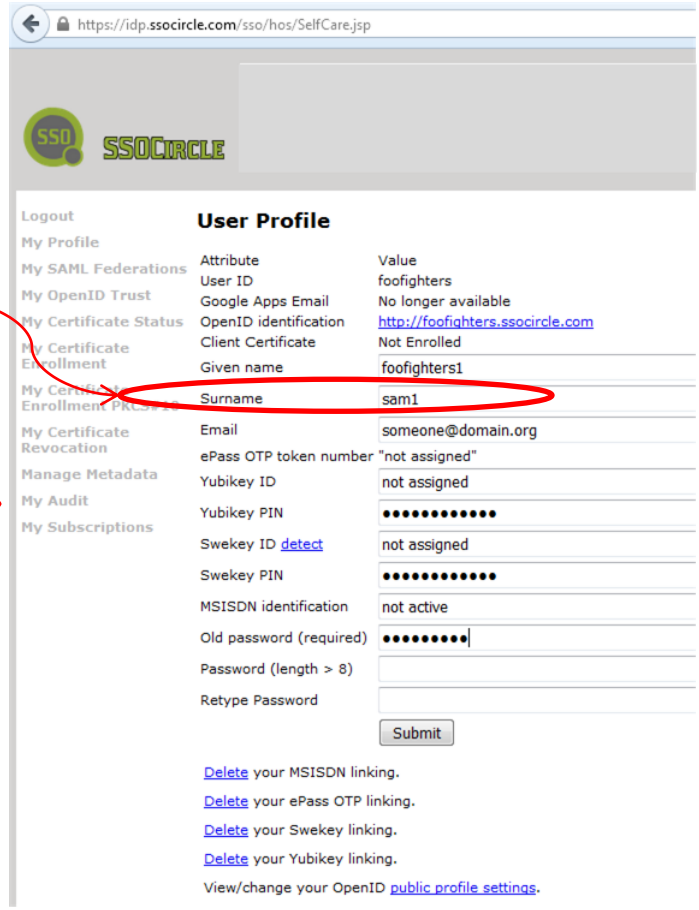
To Change Demo Users

uid=sam1,ou=People,dc=example,dc=com

DN: uid=sam1,ou=People,dc=example,dc=com

Attribute Description	Value
objectClass	***extensibleObject (auxiliary)***
objectClass	***ftMods (structural)***
objectClass	***ftProperties (structural)***
objectClass	***ftUserAttrs (structural)***
objectClass	***inetOrgPerson (structural)***
objectClass	***organizationalPerson (structural)***
objectClass	***person (structural)***
objectClass	***top (abstract)***
cn	**Sam One**
sn	**One**
description	Fortress SAML Demo User 1
displayName	Sam One
ou	org.samlsample.users
uid	sam1
userPassword	SSHA hashed password
ftCstr	*sam1$0$$$$$$$*
ftId	*a59bf210-7101-4bb9-b089-9205d594d108*
ftProps	*init:*
ftRA	*samRole1*

Change Surname field in SSO Circle Profile to use different rbac users.



Apache Fortress SAML Demo

- https://github.com/shawnmckinney/fortress-saml-demo

User to Role	Page One	Page Two	Page Three
Sam*	True	True	True
Sam1	True	False	False
Sam2	False	True	False
Sam3	False	False	True