Use Shibboleth as a CAS server

This wiki page page was contributed by John Morrison at Uppsala University.

Shibboleth IdP version 3 supports most of the CAS protocol version 2 including attribute release and CAS proxy support however this page describes the basic configuration for a normal CAS client. For a more complete information on how to configure Shibboleth CAS server please see the page CasProtocolConfiguration on the Shibboleth Wiki.

To configure Shibboleth CAS Server you need to theese steps:

CAS URI compability chart for Shibboleth CAS Server

CAS URIs	Supported
/login	Yes
/proxy	Yes, but advanced configuration
/logout	Yes
/validate	No, CAS protocol version 1
/serviceValidate	Yes
/samlValidate	Yes, but advanced configuration
/proxyValidate	Yes, but advanced configuration
/p3/serviceValidate	No, CAS protocol version 3
/p3/proxyValidate	No, CAS protocol version 3

CAS client configuration (i.e. CAS Service Provider)

The base URL for the CAS protocol on Shibboleth is https://HOSTNAME/idp/profile/cas where HOSTNAME is the DNS service name for the Shibboleth Identity Provider, for example https://idp.example.se/idp/profile/cas.

Example URL's:

- CAS login URL is https://idp.example.se/idp/profile/cas/login
- CAS logout URL is https://idp.example.se/idp/profile/cas/logout
- CAS service validate URL is https://idp.example.se/idp/profile/cas/serviceValidate

Configure CAS storage for CAS tickets and IdP sessions

In all SWAMID Shibboleth IdP configurations SWAMID suggests that JPA Storage Service is used.

- If you do not already use JPA Storage Service configure the service.
- Activate JPA Storrage Service for idp.session.StorageService and idp.cas.idp.session.StorageService in idp.properties by removing # and changing the values to shibboleth.JPAStorageService.

Update of idp.properties

```
# Set to "shibboleth.JPAStorageService" for JPA based server-side storage
of user sessions
idp.session.StorageService = shibboleth.JPAStorageService

# Storage service used by CAS protocol
# Defaults to shibboleth.StorageService (in-memory)
# MUST be server-side storage (e.g. in-memory, memcached, database)
# NOTE that idp.session.StorageService requires server-side storage
# when CAS protocol is enabled
idp.cas.StorageService = shibboleth.JPAStorageService
```

- If you use the Shibboleth Consent module, Terms of use module or High availability settings these must also use JPA Storage Service.

Configure CAS protocol settings

CAS server in Shibboleth has restrictions on which CAS clients can use the CAS functionality. To configure these restriction edit the file cas-protocol.xml. In this example the restriction is exemplified with an regex that defines all https servers within the domain example.se.

cas-protocol.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:c="http://www.springframework.org/schema/c"
       xmlns:p="http://www.springframework.org/schema/p"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
                           http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util.xsd"
       default-init-method="initialize"
       default-destroy-method="destroy">
    <!--
       | The CAS service registry defines verified relying parties by
endpoint URI.
       | The default implementation treats the ID of each entry as a
regular expression defining a logical group of
       | services whose URIs match the expression.
       |
       | This bean is reloaded periodically according to %{idp.home}/conf
/services.properties.
       -->
    <bean id="reloadableServiceRegistry"
          class="%{idp.cas.serviceRegistryClass:net.shibboleth.idp.cas.
service.PatternServiceRegistry}">
        <property name="definitions">
            <list>
               <!-- CAS is ok for all https services within the DNS domain
example.se -->
```

```
                <bean class="net.shibboleth.idp.cas.service.
ServiceDefinition"
                      c:regex="https:\/\/([A-Za-z0-9_-]+\.)*example\.se(:
\d+)?\/.*"
                      p:group="CAS-clients-in-example-se"
                      p:authorizedToProxy="false"
                      <!-- Change singleLogoutParticipant to true if you
have enabled SAML single logout! -->
                      p:singleLogoutParticipant="false" />
                  <!-- Default examples
                  <bean class="net.shibboleth.idp.cas.service.
ServiceDefinition"
                       c:regex="https://([A-Za-z0-9_-]+\.)*example\.org(:
\d+)?/.*"
                       p:group="proxying-services"
                       p:authorizedToProxy="true"
                       p:singleLogoutParticipant="true" />
                  <bean class="net.shibboleth.idp.cas.service.
ServiceDefinition"
                       c:regex="http://([A-Za-z0-9_-]+\.)*example\.org(:
\d+)?/.*"
                       p:group="non-proxying-services"
                       p:authorizedToProxy="false" /
                  -->
              </list>
          </property>
    </bean>
    <!--
       | Advanced CAS configuration.
       |
       | Override default CAS components by creating aliases to custom
components where the alias
       | is the same as the default component bean ID.
       -->
    <!--
    <bean id="cas.CustomTicketService"
          class="org.example.idp.cas.CustomTicketService" />
    <alias name="cas.CustomTicketService" alias="cas.TicketService" />
    <bean id="cas.CustomProxyAuthenticator"
          class="org.example.idp.cas.CustomProxyAuthenticator" />
    <alias name="cas.CustomProxyAuthenticator" alias="cas.
ProxyAuthenticator" />
    -->
</beans>
```

Block the generation of ePTID for CAS clients

SWAMID recommends that the attribute eduPersonTargetedID (ePTID) is released tol SAML Service Providers that don't have registered for any of the defined entity categories. CAS clients get ePTID generated for every protected web page that a user deep link to without previous CAS client login due to that the CAS client don't exist in SAML metadata and therefore hasn't any entity categories. This can be a problem if the deep links are very long.

As long as ePTID isn't used for any SAML Service Providers within you domain you can block ePTID generation for the whole DNS domain. In the example we have used a regex for https servers on the domain example.se. Please note that the same regex is used in the CAS restrictions above.

- Add a bean to block ePTID to https servers within example.se:

Add to global.xml

```
<!-- Do NOT release the eduPersonTargetedID attribute based on the REGEX --
>
<bean id="ReleaseEPTID" parent="shibboleth.Conditions.NOT">
    <constructor-arg>
        <bean parent="shibboleth.Conditions.RelyingPartyId">
            <constructor-arg>
                <!-- Never release ePTID to https servers within the
domain example.se -->
                <bean class="com.google.common.base.Predicates"
                      factory-method="containsPattern"
                      c:pattern="https:\/\/([A-Za-z0-9_-]+\.)*example\.se(:
\d+)?\/.*" />
            </constructor-arg>
        </bean>
    </constructor-arg>
</bean>
```

- Use the ReleaseEPTID bean in attribute-resolver.xml:

Add activationConditionRef="ReleaseEPTID" in attribute-resolver.xml

```
<!-- The source for this attribute is from the database StoredId and no
longer the classic computedID  -->
<resolver:AttributeDefinition xsi:type="ad:SAML2NameID"
                              id="eduPersonTargetedID"
                              nameIdFormat="urn:oasis:names:tc:SAML:2.0:
nameid-format:persistent"
                              sourceAttributeID="persistentId"
                              <!-- This will only block the release but
not the generation -->
                              activationConditionRef="ReleaseEPTID">
    <resolver:Dependency ref="StoredId" />
    <resolver:AttributeEncoder xsi:type="enc:SAML1XMLObject"
                               name="urn:oid:1.3.6.1.4.1.5923.1.1.1.10" />
    <resolver:AttributeEncoder xsi:type="enc:SAML2XMLObject"
                               name="urn:oid:1.3.6.1.4.1.5923.1.1.1.10"
                               friendlyName="eduPersonTargetedID" />
</resolver:AttributeDefinition>


<!--
    This connector relies on global.xml for the Managed connection to the
db.
    If you have a Active Directory data source change the
sourceAttributeID to sAMAccountName
-->
<resolver:DataConnector id="StoredId"
                        xsi:type="StoredId"
                        xmlns="urn:mace:shibboleth:2.0:resolver:dc"
                        generatedAttributeID="persistentId"
                        sourceAttributeID="uid"
                        salt="your random string here"
                        <!-- Important that this is here! Otherwise
generation will still happen in the database! -->
                        activationConditionRef="ReleaseEPTID">
    <resolver:Dependency ref="uid" />
    <dc:BeanManagedConnection>MyGlobalDataSource</dc:BeanManagedConnection>
</resolver:DataConnector>
```

Activate CAS protocol

The basic CAS server functionality is now configured.

- To activate the CAS server in Shibboleth you need to add two beans to the section shibboleth.DefaultRelyingParty in the file relying-party.xml.

Add CAS configuration to relying-party.xml

```
<bean id="shibboleth.DefaultRelyingParty" parent="RelyingParty">
    <property name="profileConfigurations">
        <list>
            <!-- ... -->
            <ref bean="CAS.LoginConfiguration"/>
            <ref bean="CAS.ValidateConfiguration" />
        </list>
    </property>
</bean>
```

- To startup the CAS server you need to restart Shibboleth.