IdP 3.0

Installation instructions for Shibboleth-Idp 3.1

Prerequisites (settings and configurations already within the image)

- RedHat / Centos installed with at least following packages: java-1.8.0-openjdk, ant and wget
  - yum install java-1.8.0-openjdk-headless ant wget elinks ntp mc
- Set Java 1.8 as a default JDK instead of Java 1.7 installed because ant dependencies
    - /usr/sbin/alternatives --config java
  - Set securerandom.source to urandom in /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.51-1.b16.el7_1.x86_64 /jre/lib/security/java.security
    - securerandom.source=file:/dev/./urandom

- Download latest Shibboleth Identitity Provider (Currently 3.1.2)
  - wget -qO- http://shibboleth.net/downloads/identity-provider/latest/shibboleth-identity-provider-3.1.2.tar.gz | tar xvz -C /opt

- Download latest Jetty (Currently 9.3.3) and optional logging modules
  - wget -qO- http://download.eclipse.org/jetty/stable-9/dist/jetty-distribution-9.3.3.v20150827.tar.gz | tar xvz -C /opt
  - wget -qO- http://www.slf4j.org/dist/slf4j-1.7.12.tar.gz | tar xvz -C /opt
  - wget -qO- http://logback.qos.ch/dist/logback-1.1.3.tar.gz | tar xvz -C /opt
- MariaDB
  - yum install mariadb-server
  - service mariadb start
  - chkconfig --level 2345 mariadb on
  - mysql_secure_installation
- Openldap
  - yum install openldap-servers openldap-clients
  - Modify /etc/openldap/slapd.d/cn=config/olcDatabase={2}hdb.ldif
    - olcSuffix: dc=training
    - olcRootDN: cn=Manager,dc=training
    - olcRootPW: {SSHA}rKveY28iAShYfoCobkyARJN/4TFyFK8l # (generated olcRootPW by using command slappasswd)
  - Modify /etc/openldap/slapd.d/cn=config/olcDatabase={1}monitor.ldif
    - olcAccess: {0}to * by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth" read by dn.base="cn=Manager, dc=training" read by * none
  - Disable anonymous binds
    - Add to /etc/openldap/slapd.d/cn=config/olcDatabase={-1}frontend.ldif
      - olcRequires: authc
    - Add to /etc/openldap/slapd.d/cn=config.ldif
      - olcDisallows: bind_anon
  - Enable SSL/TLS
    - Generate private key (and remove password)

        - cd /etc/pki/tls/certs/ ; make LDAP.key
        - openssl rsa -in LDAP.key -out LDAP.key
    - Generate certificate signing request
        - make LDAP.csr
    - Generate certificate
        - openssl x509 -in LDAP.csr -out LDAP.crt -req -signkey LDAP.key -days 3650
    - systemctl stop slapd
    - cp LDAP.key LDAP.crt ca-bundle.crt /etc/openldap/certs/.
    - cd /etc/openldap/certs/ ; chown ldap. LDAP.key LDAP.crt ca-bundle.crt
    - Add to /etc/openldap/slapd.d/cn=config.ldif
        - olcTLSCACertificateFile: /etc/openldap/certs/ca-bundle.crt
        - olcTLSCertificateKeyFile: /etc/openldap/certs/LDAP.key
        - olcTLSCertificateFile: /etc/openldap/certs/LDAP.crt
    - Add to /etc/openldap/slapd.d/cn=config/olcDatabase={2}hdb.ldif
        - olcSecurity: tls=1
    - add "ldaps:///" to /etc/sysconfig/slapd
        - SLAPD_URLS="ldapi:///ldap:///ldaps:///"
    - echo "TLS_REQCERT allow" >> /etc/openldap/ldap.conf
    - systemctl start slapd
- chkconfig --level 2345 slapd on
- Add schemas
    - ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/cosine.ldif
    - ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/inetorgperson.ldif
    - ldapadd -Y EXTERNAL -H ldapi:/// -f /etc/openldap/schema/nis.ldif
    - ldapadd -Y EXTERNAL -H ldapi:/// -f eduperson.ldif
    - ldapadd -Y EXTERNAL -H ldapi:/// -f schac.ldif
    - ldapadd -Y EXTERNAL -H ldapi:/// -f FunetEduPerson21.ldif

- Import base structure with ldapadd

ldapadd -x -W -Z -D "cn=Manager,dc=training"

```
dn: dc=training
objectClass: dcObject
objectClass: organization
o: csc
dc: training

dn: ou=People,dc=training
objectClass: organizationalUnit
objectClass: top
ou: People

dn: ou=Groups,dc=training
objectClass: organizationalUnit
objectClass: top
ou: Groups


dn: uid=training,ou=People,dc=training
objectClass: top
objectClass: inetOrgPerson
objectClass: funetEduPerson
objectClass: posixAccount
uid: training
uidNumber: 1
gidNumber: 65555
homeDirectory: /dev/null
schacHomeOrganization: funet.fi
schacHomeOrganizationType: urn:mace:terena.org:schac:homeOrganizationType:fi:other
eduPersonPrincipalName: training@funet.fi
givenName: training
displayName: training
cn: training account
sn: account
mail: training@funet.fi
userPassword:: dHJhaW5pbmc=
```

Installation of Jetty and Shibboleth IdP

This section goes thru installation of Jetty and Shibboleth IdP. Shibboleth will be configured as a system service owned by jetty user using jetty-setuid module. Onwards Jetty version 9.1 it is possible to separate binary installation with default configuration ${jetty.home} from own customizations ${jetty.base}. This separation makes Jetty upgrades easier and versions can be swiched with eg. symlink. Jetty installation also includes some basic hardening and optional extended logging configuration.

To access training virtual hosts used in training use following keys:

- Putty: putty.training.ppk
- OpenSSH: id_rsa_training (eg. ssh cloud-user@<HOST> -i id_rsa_training)

About pouta cluster

- https://research.csc.fi/pouta-user-guide

System configuration

Either rename unpacked Jetty directory or use symlink. With symlink method we can have multiple versions of Jetty and switching between versions during testing or upgrading is easy, (another option is to change jetty.home variable, but we dont cover it here).

Everything here is done as a root user

Change to root account

```
sudo su -
```

Create symlink

Rename jetty-distribution directory to jetty or create symlink

```
ln -s /opt/jetty-distribution-9.3.3.v20150827 /opt/jetty
```

Create and set environmental variables for Jetty. Copy paste following to your console and then load environmental variables in to the use.

/etc/default/shibboleth

```
cat << EOF > /etc/default/shibboleth
JAVA_HOME=/etc/alternatives/jre_1.8.0
JETTY_BASE=/opt/shibboleth-idp/jetty-base
JETTY_HOME=/opt/jetty
IDP_HOME=/opt/shibboleth-idp
IDP_SRC=/opt/shibboleth-identity-provider-3.1.2
export JAVA_HOME JETTY_BASE JETTY_HOME IDP_HOME IDP_SRC
EOF

source /etc/default/shibboleth
```

Install shibboleth identity provider

Install Shibboleth IdP

```
cd $IDP_SRC
./bin/install.sh
```

./bin/install.h

Accept default values except with following

Hostname: [localhost.localdomain]

Your hostname where idp is running eg: vm0401.kaj.pouta.csc.fi

SAML EntityID: [https://vm0401.kaj.pouta.csc.fi/idp/shibboleth]

Entity id with hostname part, workshop identifier and your team identifier eg: https://vm401.kaj.pouta.csc.fi/ShibIdPv3WorkShop/SSi

Attribute Scope: [localdomain]

Scope used with scoped attributes eg: funet.fi

Passwords you give for TLS Private Key and Cookie Encryption Key are used by shibboleth (/opt/shibboleth-idp/conf/idp.properties)

Create jetty-base, jetty user and keystore

Copy default jetty-base under $IDP_HOME with jetty-ssl-context.xml

Copy jetty-base under shibboleth with jetty-ssl-context.xml

```
cp -R $IDP_SRC/jetty-base $IDP_HOME/jetty-base
cp $JETTY_HOME/etc/jetty-ssl-context.xml $JETTY_BASE/etc/jetty-ssl-context.xml
```

Create Jetty user and group

Required for setuid functionality

```
useradd --user-group --shell /bin/false --home-dir $JETTY_BASE/tmp jetty
usermod -a -G jetty jetty
```

The useradd command gives you warning about the home directory which already exists. $JETTY_BASE/tmp directory already exists and you dont need to have skel contents in there so it's ok and you can continue and ignore warning.

Create keystore for jetty, there are already keystores but we will create new one for a training purposes. Keystore is used by jetty https handler and it contains keys and certificates. Keystore is password protected and at the next section you have to insert keystore location and passwords for opening the keystore.

generate keystore with keytool

```
keytool -keystore $IDP_HOME/credentials/keystore.jks -alias jetty -genkey -keyalg RSA
```

keytool -keystore $IDP_HOME/credentials/keystore.jks -alias jetty -genkey -keyalg RSA

Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: Sami Silén
What is the name of your organizational unit?
[Unknown]: IAM
What is the name of your organization?
[Unknown]: CSC
What is the name of your City or Locality?
[Unknown]: Espoo
What is the name of your State or Province?
[Unknown]: Uusimaa
What is the two-letter country code for this unit?
[Unknown]: FI
Is CN=Sami Silén, OU=IAM, O="CSC ", L=Espoo, ST=Uusimaa, C=FI correct?
[no]: yes

Enter key password for <jetty>
(RETURN if same as keystore password):

Set proper ownership for directories and credential files

Required for setuid functionality

```
chown jetty. $IDP_HOME/logs $JETTY_BASE -R
chown root:jetty $IDP_HOME/conf $IDP_HOME/credentials -R
chmod 750 $IDP_HOME/conf $IDP_HOME/credentials -R
```

Configure Jetty

Edit $JETTY_BASE/start.d/idp.ini

```
jetty.host=0.0.0.0

jetty.https.port=443
jetty.backchannel.port=8443

jetty.backchannel.keystore.path=../credentials/keystore.jks
jetty.browser.keystore.path=../credentials/keystore.jks

jetty.backchannel.keystore.password=<PASSWORD>
jetty.browser.keystore.password=<PASSWORD>

jetty.backchannel.keystore.type=JKS
jetty.browser.keystore.type=JKS
```

For setuid (running shibboleth under jetty user) uncomment setuid from start.ini.

$JETTY_BASE/start.ini

```
--module=setuid
```

Harden Jetty SSL by removing unsecure protocols and ciphers. Replace <Set name="ExcludeCipherSuites"> section with following. This is an example so if you have time you can play with "https://www.ssllabs.com/ssltest/" and see what more we're needing for grade A.

$JETTY_BASE/etc/jetty-ssl-context.xml

```
<Set name="excludeProtocols">
        <Array type="String">
            <Item>SSLv3</Item>
    </Array>
</Set>
<Set name="IncludeCipherSuites">
        <Array type="String">
            <Item>TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256</Item>
        <Item>TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384</Item>
        <Item>TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256</Item>
        <Item>TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384</Item>
        <Item>TLS_RSA_WITH_AES_128_GCM_SHA256</Item>
        <Item>TLS_RSA_WITH_AES_256_GCM_SHA256</Item>
        <Item>TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256</Item>
        <Item>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384</Item>
        <Item>TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA</Item>
        <Item>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA</Item>
        <Item>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256</Item>
        <Item>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384</Item>
        <Item>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA</Item>
        <Item>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA</Item>
        <Item>TLS_RSA_WITH_AES_128_CBC_SHA256</Item>
        <Item>TLS_RSA_WITH_AES_256_CBC_SHA384</Item>
        <Item>TLS_RSA_WITH_AES_128_CBC_SHA</Item>
        <Item>TLS_RSA_WITH_AES_256_CBC_SHA</Item>
        </Array>
</Set>
```

Enable logback for all Jetty logging <optional>

Copy needed files under your JETTY_BASE and update logback if necessary. Shibboleth already provides needed logback and slf4j libraries, in here we just upgrade those to the latest releases.

files under $JETTY_BASE/lib/logging

```
rm $JETTY_BASE/lib/logging/logback*
cp /opt/logback-1.1.3/logback-*3.jar $JETTY_BASE/lib/logging/.
cp /opt/slf4j-1.7.12/slf4j-api-1.7.12.jar $JETTY_BASE/lib/logging/.
chown jetty. $JETTY_BASE/lib/logging/*
```

jetty-base under /opt/shibboleth-identity-provider-3.1.2 already contained configuration for logback, you can check that jetty-requestlog.xml contains pointer to logback.

$JETTY_BASE/etc/jetty-requestlog.xml

```
<Set name="fileName"><Property name="jetty.base" default="." />/resources/logback-access.xml</Set>
```

Enable shibboleth as a systemd service

Create systemd configuration file /etc/systemd/system/shibboleth.service

/etc/systemd/system/shibboleth.service

```
cat << EOF > /etc/systemd/system/shibboleth.service
[Unit]
Description=Shibboleth Identity Provider

[Service]
EnvironmentFile=-/etc/default/shibboleth
WorkingDirectory=/opt/shibboleth-idp/jetty-base
ExecStart=/usr/bin/java -jar /opt/jetty/start.jar jetty.home=/opt/jetty jetty.base=/opt/shibboleth-idp/jetty-
base

[Install]
WantedBy=multi-user.target
EOF
```

Reload systemd configuration and enable shibboleth service before starting shibboleth, we dont yet start shibboleth via systemctl.

Systemctl configuration

```
systemctl daemon-reload
systemctl enable shibboleth
# systemctl start shibboleth
```

Deliverables

You should be able to start your IdP and test functionality

Test your installation

```
cd /opt/shibboleth-idp/jetty-base; /usr/bin/java -jar /opt/jetty/start.jar jetty.home=/opt/jetty jetty.base=/opt
/shibboleth-idp/jetty-base

# depending of a logging configuration you might not see any output on the screen #
# you can check log files:
#  * /opt/shibboleth-idp/logs/idp-process.log
#  * /opt/shibboleth-idp/jetty-base/logs/jetty.log
#  * /opt/shibboleth-idp/jetty-base/logs/2015_11_02.stderrout.log  (choose latest one)

SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/jetty-distribution-9.3.3.v20150827/lib/logging/logback-classic-1.1.3.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/shibboleth-idp/jetty-base/lib/logging/logback-classic-1.1.2.jar!/org
/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [ch.qos.logback.classic.util.ContextSelectorStaticBinder]
```

*) If you are facing problems with starting Jetty (Unsupported major.minor version 52.0), check that your linked java version is 1.8 with "/usr/sbin/alternatives --config java"

You should now be able to connect your jetty instance via browser using https

https://<your instance>/idp

```
Our Identity Provider

(Replace this placeholder with your
organizational logo / label)

No services are available at this location.
```

Installation phase is now finished

Now you can break installation testing with ctrl+break and continue with shibboleth IdP configuration. Now you can start and stop you shibboleth idp with systemctl commands

systemctl start shibboleth

```
systemctl start shibboleth
```

Shibboleth IdP configuration

Metadata provider configuration

MetadataProvider configuration is moved to own separate file "metadata-providers.xml" (old: relaying-part.xml).
When configuring MetadataFilter for the MetadataProvider you don't need anymore trustEngineRef attribute for referencing trust engine, instead you can define certificate with the certificateFile attribute.

/opt/shibboleth-idp/conf/metadata-providers.xml

```
<!-- Metadata is refreshed every hour -->
<MetadataProvider id="HTTPMetadata" xsi:type="FileBackedHTTPMetadataProvider" refreshDelayFactor="0.5"
maxRefreshDelay="PT2H" httpCaching="memory"
  backingFile="%{idp.home}/metadata/backingFiles/haka_test_metadata_signed.xml"
  metadataURL="https://haka.funet.fi/metadata/haka_test_metadata_signed.xml">

  <MetadataFilter xsi:type="SignatureValidation" certificateFile="%{idp.home}/credentials/haka_testi_2015_sha2.
crt" />
  <MetadataFilter xsi:type="EntityRoleWhiteList">
    <RetainedRole>md:SPSSODescriptor</RetainedRole>
  </MetadataFilter>
</MetadataProvider>
```

Check that backingFile directory exists and jetty has rights to write under it.

```
mkdir -p  $IDP_HOME/metadata/backingFiles
chown -R jetty. $IDP_HOME/metadata
```

Download Signature validation certificate

```
 wget -O $IDP_HOME/credentials/haka_testi_2015_sha2.crt 'https://confluence.csc.fi/download/attachments/31195585
/haka_testi_2015_sha2.crt?version=1&modificationDate=1430212953940&api=v2'
```

Configure IdP to use LDAP authentication

Password authentication is in use by default. You can double check from ***idp.properties*** that there is definition " idp.authn.flows= Password". LDAP is just one of the possible password authentication back-ends supported by IdP. We need to make sure ldap is in use and not something else like kerberos or jaas. That is configured in **password-authn-config.xml** file. make sure that correct backend is in use "<import resource="ldap-authn-config.xml" />"

LDAP properties are conveniently stored in file **ldap.properties** . You need to make following modifications for the provided example configuration.

/opt/shibboleth-idp/conf/ldap.properties

```
...
idp.authn.LDAP.authenticator                   = bindSearchAuthenticator
...
idp.authn.LDAP.ldapURL                          = ldap://localhost:389
..
idp.authn.LDAP.trustCertificates                = /etc/pki/tls/certs/LDAP.crt
...
idp.authn.LDAP.baseDN                           = ou=people,dc=training
...
idp.authn.LDAP.bindDN                           = uid=training,ou=people,dc=training
idp.authn.LDAP.bindDNCredential                 = <PASSWORD>
```

Configure attribute resolver

Shibboleth comes with attribute-resolver-ldap.xml and attribute-resolver-full.xml which you can use as a starting point and construct your own resolver according to your backend and attribute locations.

Default ldap configuration should work out of the box if you have configure ldap.properties file above, then you can continue and go thru attributes you are going to release, you should also set friendly names for attributes. Friendly names are used by consent module (configured later in section: "Configure consent module").

$IDP_HOME/conf/attribute-resolver.xml

```
...
   <resolver:AttributeDefinition xsi:type="ad:Prescoped" id="eduPersonPrincipalName" sourceAttributeID="
eduPersonPrincipalName">
        <resolver:Dependency ref="myLDAP" />
        <resolver:DisplayName xml:lang="fi">Henkilön yksilöivä tunniste</resolver:DisplayName>
        <resolver:DisplayName xml:lang="en">Principal name</resolver:DisplayName>
        <resolver:DisplayName xml:lang="se"></resolver:DisplayName>
        <resolver:DisplayDescription xml:lang="fi">Erottaa henkilön muista käyttäjistä. Muotoa
"käyttäjätunnus@domain".</resolver:DisplayDescription>
        <resolver:DisplayDescription xml:lang="en">The "NetID" of the person for the purposes of inter-
institutional authentication. Represented in the form "user@scope" where scope defines a local security domain.<
/resolver:DisplayDescription>
        <resolver:DisplayDescription xml:lang="se"></resolver:DisplayDescription>
        <resolver:AttributeEncoder xsi:type="enc:SAML1ScopedString" name="urn:mace:dir:attribute-def:
eduPersonPrincipalName" encodeType="false" />
        <resolver:AttributeEncoder xsi:type="enc:SAML2ScopedString" name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
friendlyName="eduPersonPrincipalName" encodeType="false" />
    </resolver:AttributeDefinition>
...
```

Configure attribute filter

In the attribute filter you describe which attributes gets filtered out, This example passes nameidattr for everyone, this is needed for persistentID generation (read section "Configure database storage for consent module and persistentID"). The rest attributes are released if those are requested by SP via metadata.

$IDP_HOME/conf/attribute-filter.xml

```
<afp:AttributeFilterPolicyGroup id="ShibbolethFilterPolicy"
        xmlns:afp="urn:mace:shibboleth:2.0:afp"
        xmlns:basic="urn:mace:shibboleth:2.0:afp:mf:basic"
        xmlns:saml="urn:mace:shibboleth:2.0:afp:mf:saml"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:mace:shibboleth:2.0:afp http://shibboleth.net/schema/idp/shibboleth-afp.xsd
                            urn:mace:shibboleth:2.0:afp:mf:basic http://shibboleth.net/schema/idp/shibboleth-
afp-mf-basic.xsd
                            urn:mace:shibboleth:2.0:afp:mf:saml http://shibboleth.net/schema/idp/shibboleth-afp-
mf-saml.xsd">

    <afp:AttributeFilterPolicy id="releaseNameIdToAnyone">
      <afp:PolicyRequirementRule xsi:type="basic:ANY"/>
      <afp:AttributeRule attributeID="nameidattr">
        <afp:PermitValueRule xsi:type="basic:ANY"/>
      </afp:AttributeRule>
    </afp:AttributeFilterPolicy>

    <afp:AttributeFilterPolicy id="Haka">
      <afp:PolicyRequirementRule xsi:type="basic:ANY" />
      <afp:AttributeRule attributeID="eduPersonTargetedID">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="eduPersonPrincipalName">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="eduPersonAffiliation">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="eduPersonOrgUnitDN">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="eduPersonPrimaryAffiliation">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="eduPersonPrimaryOrgUnitDN">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="schacHomeOrganization">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="schacHomeOrganizationType">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="schacDateOfBirth">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="schacPersonalUniqueCode">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="schacPersonalUniqueID">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="mail">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="uid">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="sn">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="cn">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="displayName">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="givenName">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
```

```
      <afp:AttributeRule attributeID="employeeNumber">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="mobile">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="o">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="preferredLanguage">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="telephoneNumber">
        <afp:PermitValueRule xsi:type="saml:AttributeInMetadata" onlyIfRequired="false"/>
      </afp:AttributeRule>
    </afp:AttributeFilterPolicy>
</afp:AttributeFilterPolicyGroup>
```

Configure database storage for consent module and persistentID

Configure JPAStorageService beans to the $IDP_HOME/conf/global.xml

$IDP_HOME/conf/global.xml

```
<bean id="shibboleth.JPAStorageService" class="org.opensaml.storage.impl.JPAStorageService"
             p:cleanupInterval="%{idp.storage.cleanupInterval:PT10M}"
         c:factory-ref="shibboleth.JPAStorageService.EntityManagerFactory" />

<bean id="shibboleth.JPAStorageService.EntityManagerFactory" class="org.springframework.orm.jpa.
LocalContainerEntityManagerFactoryBean">
      <property name="persistenceUnitName" value="storageservice" />
      <property name="packagesToScan" value="org.opensaml.storage.impl" />
      <property name="dataSource" ref="shibboleth.JPAStorageService.DataSource" />
      <property name="jpaVendorAdapter" ref="shibboleth.JPAStorageService.JPAVendorAdapter" />
      <property name="jpaDialect">
        <bean class="org.springframework.orm.jpa.vendor.HibernateJpaDialect" />
      </property>
</bean>

<!-- MariaDB configuration -->
<bean id="shibboleth.JPAStorageService.JPAVendorAdapter" class="org.springframework.orm.jpa.vendor.
HibernateJpaVendorAdapter">
      <property name="database" value="MYSQL" />
</bean>
<bean id="shibboleth.JPAStorageService.DataSource" class="com.zaxxer.hikari.HikariDataSource" destroy-method="
close" lazy-init="true"
          p:driverClassName="org.mariadb.jdbc.Driver"
          p:jdbcUrl="jdbc:mariadb://localhost:3306/training?autoReconnect=true&amp;
sessionVariables=wait_timeout=31536000"
          p:username="<USERNAME>"
          p:password="<PASSWORD>" />
```

This configuration uses mariadb and HikariDatasource therefore you have to install latest drivers mariadb-java-client-x.x.x.jar and HikariCP-x.x.x.jar to the $IDP_HOME /webapp/WEB-INF/lib. You have to download these from the internet

- HikariCP : https://brettwooldridge.github.io/HikariCP/
- MariaDB : https://downloads.mariadb.org/connector-java/

Configure desired modules to use JPA Storage service in `$IDP_HOME/idp.properties`.

$IDP_HOME/idp.properties

```
idp.session.StorageService=shibboleth.JPAStorageService
idp.consent.StorageService=shibboleth.JPAStorageService
# next ones are skipped on purpose, we dont put those in the mariaDB
# idp.replayCache.StorageService = shibboleth.JPAStorageService
# idp.artifact.StorageService = shibboleth.JPAStorageService
```

Currently there is an bug in shibboleth IdP so you cannot set idp.persistentId.store to use directly shibboleth.JPAStorageService and you have to make wrapper for it. First configure idp.persistentId.store to use eg: myPersistentStore

$IDP_HOME/saml-nameid.properties

```
idp.persistentId.store = myPersistentIdStore
idp.persistentId.generator = shibboleth.StoredPersistentIdGenerator
```

Then create wrapper for myPersistentStore in saml-nameid.xml (eg: straight before "SAML 2 NameID Generation" section)

$IDP_HOME/saml-nameid.xml

```
<!-- Store for persistent IDs -->
<bean id="myPersistentIdStore" class="net.shibboleth.idp.saml.nameid.impl.JDBCPersistentIdStore">
  <property name="dataSource" ref="shibboleth.JPAStorageService.DataSource" />
</bean>
```

Uncomment shibboleth.SAML2PersistentGenerator

$IDP_HOME/saml-nameid.xml

```
<!-- Uncommenting this bean requires configuration in saml-nameid.properties. -->
        <ref bean="shibboleth.SAML2PersistentGenerator" />
```

Configure persistentID

In the attirubte-filter.xml we already made an rule to pass nameidattr. We're using uid as a seed for persistentID so in the attribute-resolver.xml we configure source for nameidattr to be an uid attribute from myLDAP.

$IDP_HOME/conf/attribute-resolver.xml

```
<resolver:AttributeDefinition xsi:type="ad:Simple" id="nameidattr" sourceAttributeID="uid">
  <resolver:Dependency ref="myLDAP"/>
</resolver:AttributeDefinition>
```

In the saml-nameid.properties we now configure IdP to use this attribute for the persistentID generation as well as random SALT

$IDP_HOME/conf/saml-nameid.properties

```
idp.persistentId.sourceAttribute = nameidattr
idp.persistentId.salt = <RANDOM SALT>
```

In the c14/subject-c14n.xml uncomment SAML2Persistent strategy.

$IDP_HOME/conf/c14/subject-c14n.xml

```
<!-- Handle a SAML 2 persistent ID, provided a stored strategy is in use. -->
<ref bean="c14n/SAML2Persistent" />
```

Now because we defined nameidattr we have to hide from the enduser for avoiding confusion. Blacklist nameidattr like transientId, persistentId and eduPersonTargetedId. Add nameidattr to list of BlacklistedAttributeIDs in consent module.

$IDP_HOME/conf/intercept/consent-intercept-config.xml

```
<util:list id="shibboleth.consent.attribute-release.BlacklistedAttributeIDs">
  <value>nameidattr</value>
  <value>transientId</value>
  <value>persistentId</value>
  <value>eduPersonTargetedID</value>
</util:list>
```

Configure consent module

Consent module is on by default so it only needs customization.

$IDP_HOME/conf/idp.properties

```
# Storage service should be already configured, because we did it earlier.
idp.consent.StorageService = shibboleth.JPAStorageService
# We want reconfirmation from the user if set of released attributes changes.
idp.consent.compareValues = true
# No limits for the stored consents.
idp.consent.maxStoredRecords = -1
```

Configure default relying party

For default configuration we want to do some changes, we disable assertion encryption and response signing, but we want to sign assertions.

$IDP_HOME/conf/relaying-party.xml

```
<bean id="shibboleth.DefaultRelyingParty" parent="RelyingParty">
  <property name="profileConfigurations">
    <list>
      <bean parent="Shibboleth.SSO" p:postAuthenticationFlows="attribute-release" />
      <ref bean="SAML1.AttributeQuery" />
      <ref bean="SAML1.ArtifactResolution" />
      <bean parent="SAML2.SSO"
        p:signResponses="false"
        p:signAssertions="true"
        p:encryptAssertions="false"
        p:postAuthenticationFlows="attribute-release" />
      <ref bean="SAML2.ECP" />
      <ref bean="SAML2.Logout" />
      <ref bean="SAML2.AttributeQuery" />
      <ref bean="SAML2.ArtifactResolution" />
    </list>
  </property>
</bean>
```