

Installing and configuring Shibboleth as the backend IDP server for use with EFT SSO

THE INFORMATION IN THIS ARTICLE APPLIES TO:

- EFT v7.3.3 and later

NOTE: This article provides guidelines for using a third-party tool with EFT. This article is not meant as formal support for that tool, but only as an example of setup options. Contact the third-party seller's support for detailed information about their product. Globalscape is not responsible for any configuration errors involving the third-party tool.

Overview

- This document outlines the steps needed to install and configure Shibboleth as the backend IDP server for use with testing EFT's SSO Feature.
- There are several ways to configure shibboleth for testing this covers one very basic installation that authenticates users against an LDAP server.
- Shibboleth backend that authenticates against an LDAP server
- Shibboleth and ADFS are the most ideal test IDPs for long term EFT Automation testing in that the other test servers (safenet, salesforce) are reside in the cloud outside of our control. Shibboleth however requires the most setup time.

Shibboleth Installation

- Download Shibboleth Identity provider for Windows.
- Follow the basic installation instructions and tests to verify successful install. If you get service unavailable or unauthorized add your IP address to the attribute-filter.xml file located at shibboleth\idp\conf
- <https://wiki.shibboleth.net/confluence/display/IDP30/Installation>

Starting and Stopping Shibboleth

- To restart the shibboleth service locate the shibd_idpw.exe application located at ..\shibboleth\idp\bin\ . The executable contains service restart buttons.
- After restarting the shibboleth service you will also need to restart the jetty web server application start.jar located at ..\shibboleth\jetty

You may want to make shortcut buttons to the aforementioned services since you will likely need to restart them often.

Shibboleth Configuration

ldap.properties

Below is an excerpt of notable configuration items from the ldap.properties file located at ..\Shibboleth\idp\conf

- You will need to edit the ldap.properties file with your LDAP configuration that Shibboleth will be authenticating against.
- Open up the ldap.properties file located at ..\Shibboleth\IDP\conf
- Configure the connection properties.
- In the example below, SSL settings are set to false, if you choose to use SSL and set useSSL to true then you will need to obtain your LDAP server's certificate information and and the SSL settings below.
- Configure the DN Resolution Properties,
- Configure the Attribute Resolver properties.

```
idp.authn.LDAP.authenticator= adAuthenticator

## Connection properties ##
idp.authn.LDAP.ldapURL= ldap://192.168.100.131:389
idp.authn.LDAP.useStartTLS           = false
idp.authn.LDAP.useSSL                = false
idp.authn.LDAP.connectTimeout        = 3000

## SSL configuration, either jvmTrust, certificateTrust, or keyStoreTrust
#idp.authn.LDAP.sslConfig              = certificateTrust
## If using certificateTrust above, set to the trusted certificate's path
# idp.authn.LDAP.trustCertificates= %{idp.home}/credentials/ldap-server.crt
## If using keyStoreTrust above, set to the truststore path
##idp.authn.LDAP.trustStore= %{idp.home}/credentials/ldap-server.truststore

## Return attributes during authentication
idp.authn.LDAP.returnAttributes= passwordExpirationTime,loginGraceRemaining

## DN resolution properties ##

idp.authn.LDAP.baseDN=
idp.authn.LDAP.userFilter= (objectClass=person)
idp.authn.LDAP.bindDN=
idp.authn.LDAP.bindDNCredential=

idp.authn.LDAP.dnFormat= %s@qatappin.local

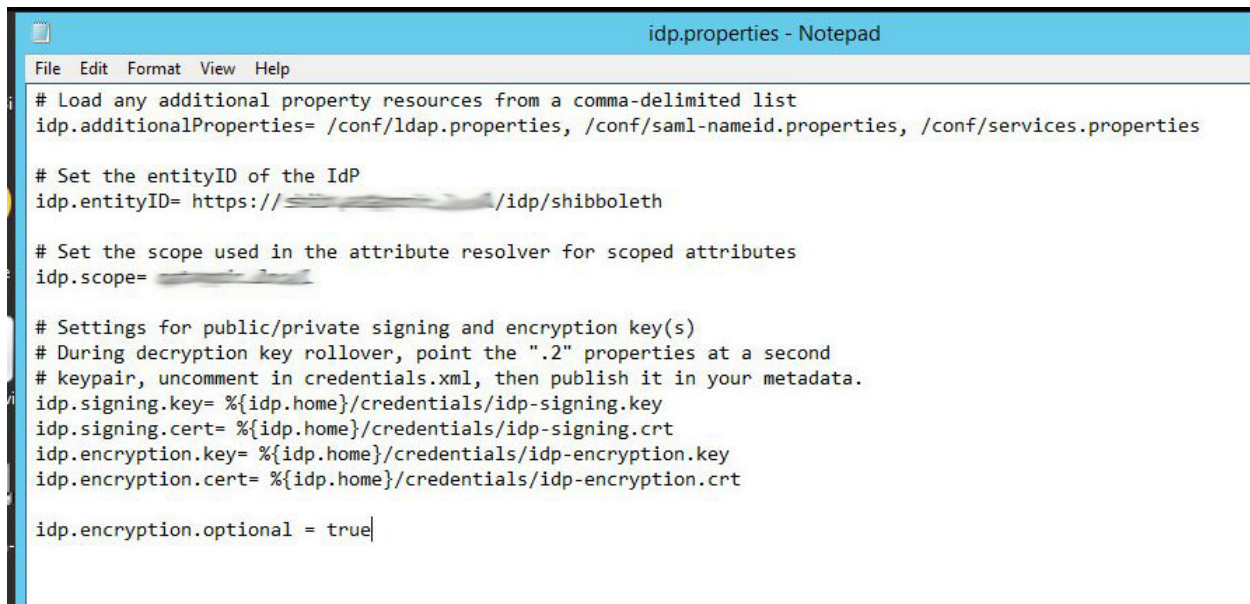
# LDAP attribute configuration, see attribute-resolver.xml
# Note, this likely won't apply to the use of legacy V2 resolver configurations
idp.attribute.resolver.LDAP.ldapURL= %{idp.authn.LDAP.ldapURL}
idp.attribute.resolver.LDAP.baseDN= %{idp.authn.LDAP.baseDN:undefined}
idp.attribute.resolver.LDAP.bindDN= %{idp.authn.LDAP.bindDN:undefined}
idp.attribute.resolver.LDAP.bindDNCredential= %{idp.authn.LDAP.bindDNCredential:undefined}
idp.attribute.resolver.LDAP.useStartTLS= %{idp.authn.LDAP.useStartTLS:true}
idp.attribute.resolver.LDAP.trustCertificates= %{idp.authn.LDAP.trustCertificates:undefined}
idp.attribute.resolver.LDAP.searchFilter= (uid=$resolutionContext.principal)
idp.attribute.resolver.LDAP.returnAttributes= cn,homephone,mail
```

Installing and configuring Shibboleth as the backend IDP server for use with EFT SSO

idp.properties

Below is an excerpt of notable configuration items from the idp.properties file located at ..\Shibboleth\idp\conf

- Take note of the idp.additionalProperties field, this field allows you to specify other properties files that will be imported into the shibboleth runtime.
- The idp.entityid can be found here, this value will be needed when configuring EFT's SSO Connection properties.
- Set the idp.scope equal to your domain.
- The location of the idp-signing.crt is also specified here. This certificate will need to be copied to your EFT System and specified in the SSO Configuration.



```
idp.properties - Notepad
File Edit Format View Help
# Load any additional property resources from a comma-delimited list
idp.additionalProperties= /conf/ldap.properties, /conf/saml-nameid.properties, /conf/services.properties

# Set the entityID of the IdP
idp.entityID= https://[redacted]/idp/shibboleth

# Set the scope used in the attribute resolver for scoped attributes
idp.scope= [redacted]

# Settings for public/private signing and encryption key(s)
# During decryption key rollover, point the ".2" properties at a second
# keypair, uncomment in credentials.xml, then publish it in your metadata.
idp.signing.key= %{idp.home}/credentials/idp-signing.key
idp.signing.cert= %{idp.home}/credentials/idp-signing.crt
idp.encryption.key= %{idp.home}/credentials/idp-encryption.key
idp.encryption.cert= %{idp.home}/credentials/idp-encryption.crt

idp.encryption.optional = true
```

relying-party.xml

- For this example configuration, edit the rely-ing-party.xml file located in the config directory to signResponses, signEncryptions but to not encryptAssertions for communication with our EFT SP.

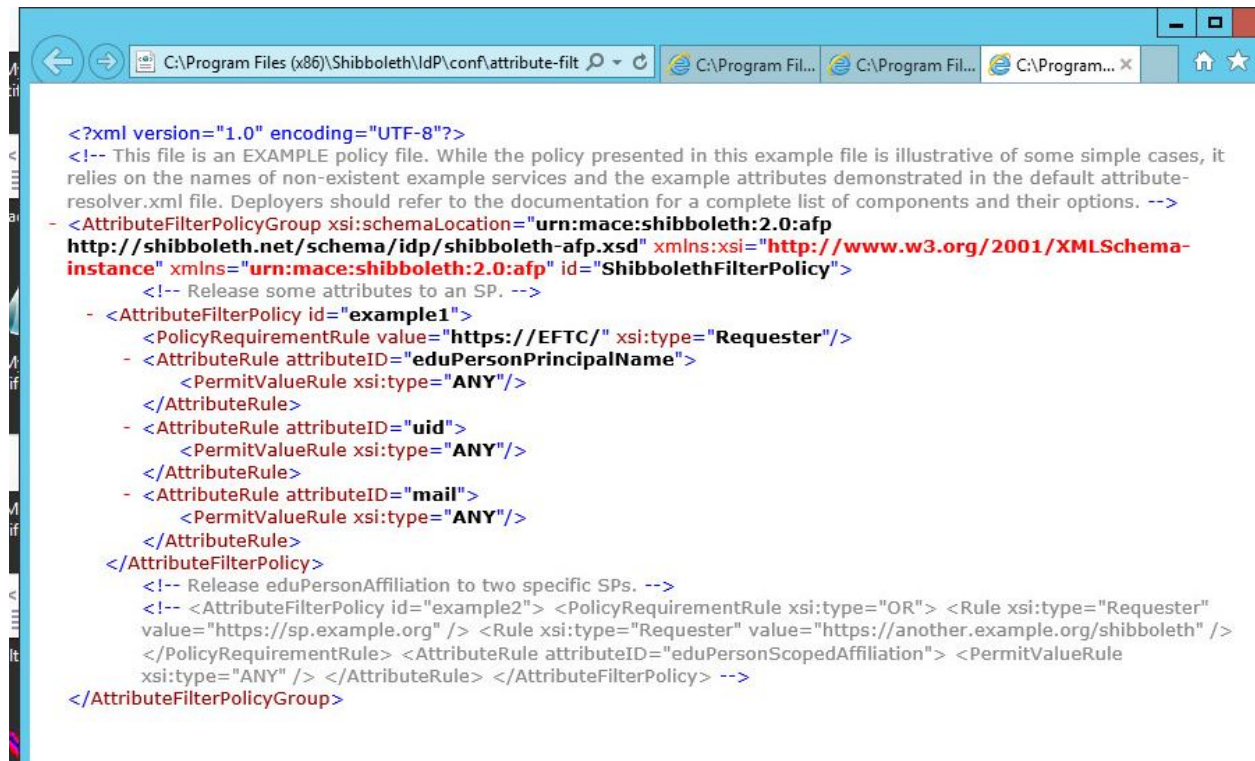


Mapping EFT Attributes to IDP > LDAP backend

The attribute-filter, attribute-resolver and attribute-resolver-ldap files contain the attribute mapping/translations. In EFT we will supply a parameter attribute/nameid to the IDP. From there we need to define a mapping of the EFT attribute to the idp server to the ldap server.

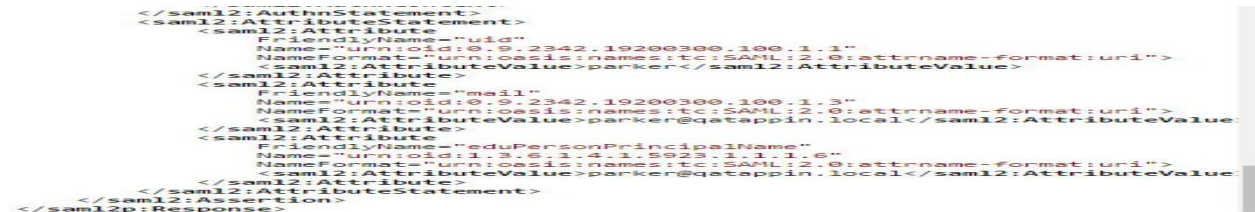
Attribute-filter.xml

- This file determines which attributes we will return to the SP (EFT). Below if you sniff the SAML Response you will see the attributes defined in the attribute-filter.xml document.



```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This file is an EXAMPLE policy file. While the policy presented in this example file is illustrative of some simple cases, it
relies on the names of non-existent example services and the example attributes demonstrated in the default attribute-
resolver.xml file. Deployers should refer to the documentation for a complete list of components and their options. -->
- <AttributeFilterPolicyGroup xsi:schemaLocation="urn:mace:shibboleth:2.0:afp
http://shibboleth.net/schema/idp/shibboleth-afp.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns="urn:mace:shibboleth:2.0:afp" id="ShibbolethFilterPolicy">
  <!-- Release some attributes to an SP. -->
  - <AttributeFilterPolicy id="example1">
    <PolicyRequirementRule value="https://EFTC/" xsi:type="Requester"/>
    - <AttributeRule attributeID="eduPersonPrincipalName">
      <PermitValueRule xsi:type="ANY"/>
    </AttributeRule>
    - <AttributeRule attributeID="uid">
      <PermitValueRule xsi:type="ANY"/>
    </AttributeRule>
    - <AttributeRule attributeID="mail">
      <PermitValueRule xsi:type="ANY"/>
    </AttributeRule>
  </AttributeFilterPolicy>
  <!-- Release eduPersonAffiliation to two specific SPs. -->
  <!-- <AttributeFilterPolicy id="example2"> <PolicyRequirementRule xsi:type="OR"> <Rule xsi:type="Requester"
value="https://sp.example.org" /> <Rule xsi:type="Requester" value="https://another.example.org/shibboleth" />
</PolicyRequirementRule> <AttributeRule attributeID="eduPersonScopedAffiliation"> <PermitValueRule
xsi:type="ANY" /> </AttributeRule> </AttributeFilterPolicy> -->
</AttributeFilterPolicyGroup>
```

- Below Image capture of a SAML Response to an EFT SP request:



```
</saml2:AuthnStatement>
<saml2:AttributeStatement>
  <saml2:Attribute
    FriendlyName="uid"
    Name="urn:oid:0.9.2342.19200300.100.1.1"
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
    <saml2:AttributeValue>parker</saml2:AttributeValue>
  </saml2:Attribute>
  <saml2:Attribute
    FriendlyName="mail"
    Name="urn:oid:0.9.2342.19200300.100.1.3"
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
    <saml2:AttributeValue>parker@qatappin.local</saml2:AttributeValue>
  </saml2:Attribute>
  <saml2:Attribute
    FriendlyName="eduPersonPrincipalName"
    Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
    <saml2:AttributeValue>parker@qatappin.local</saml2:AttributeValue>
  </saml2:Attribute>
</saml2:AttributeStatement>
</saml2:Assertion>
</saml2p:Response>
```

Installing and configuring Shibboleth as the backend IDP server for use with EFT SSO

Attribute-resolver.xml

```
xmlns:resolver="urn:mace:shibboleth:2.0:resolver">
<!-- Attribute Definitions -->
<!-- The EPPN is the "standard" federated username in higher ed. For guidelines on the implementation of this attribute, refer to the Shibboleth and
eduPerson documentation. Above all, do not expose a value for this attribute without considering the long term implications. -->
- <resolver:AttributeDefinition sourceAttributeID="uid" scope="{idp.scope}" xsi:type="ad:Scoped" id="eduPersonPrincipalName">
  <resolver:Dependency ref="uid"/>
  <resolver:AttributeEncoder xsi:type="enc:SAML1ScopedString" encodeType="false" name="urn:mace:dir:attribute-
def:eduPersonPrincipalName"/>
  <resolver:AttributeEncoder xsi:type="enc:SAML2ScopedString" encodeType="false" name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
friendlyName="eduPersonPrincipalName"/>
</resolver:AttributeDefinition>
<!-- The uid is the closest thing to a "standard" LDAP attribute representing a local username, but you should generally "never" expose uid to federated
services, as it is rarely globally unique. -->
- <resolver:AttributeDefinition xsi:type="ad:PrincipalName" id="uid">
  <resolver:AttributeEncoder xsi:type="enc:SAML1String" encodeType="false" name="urn:mace:dir:attribute-def:uid"/>
  <resolver:AttributeEncoder xsi:type="enc:SAML2String" encodeType="false" name="urn:oid:0.9.2342.19200300.100.1.1" friendlyName="uid"/>
</resolver:AttributeDefinition>
<!-- In the rest of the world, the email address is the standard identifier, despite the problems with that practice. Consider making the EPPN value the
same as your official email addresses whenever possible. -->
- <resolver:AttributeDefinition xsi:type="ad:Template" id="mail">
  <resolver:Dependency ref="uid"/>
  <resolver:AttributeEncoder xsi:type="enc:SAML1String" encodeType="false" name="urn:mace:dir:attribute-def:mail"/>
  <resolver:AttributeEncoder xsi:type="enc:SAML2String" encodeType="false" name="urn:oid:0.9.2342.19200300.100.1.3" friendlyName="mail"/>
  - <ad:Template>
    <![CDATA[ ${uid}@qatappin.local ]]>
  </ad:Template>
  <ad:SourceAttribute>uid</ad:SourceAttribute>
</resolver:AttributeDefinition>
<!-- This is an example of an attribute sourced from a data connector. -->
- <resolver:AttributeDefinition sourceAttributeID="affiliation" scope="{idp.scope}" xsi:type="ad:Scoped" id="eduPersonScopedAffiliation">
  <resolver:Dependency ref="staticAttributes"/>
  <resolver:AttributeEncoder xsi:type="enc:SAML1ScopedString" encodeType="false" name="urn:mace:dir:attribute-
def:eduPersonScopedAffiliation"/>
  <resolver:AttributeEncoder xsi:type="enc:SAML2ScopedString" encodeType="false" name="urn:oid:1.3.6.1.4.1.5923.1.1.1.9"
friendlyName="eduPersonScopedAffiliation"/>
</resolver:AttributeDefinition>
<!-- Data Connectors -->
- <resolver:DataConnector xsi:type="dc:Static" id="staticAttributes">
  - <dc:Attribute id="affiliation">
    <dc:Value>member</dc:Value>
  </dc:Attribute>
</resolver:DataConnector>
</resolver:AttributeResolver>
```

Attribute-resolver-ldap.xml

```
xmlns:resolver="urn:mace:shibboleth:2.0:resolver"
xmlns:security="urn:mace:shibboleth:2.0:security"
xmlns:enc="urn:mace:shibboleth:2.0:encoder"
xmlns:ad="urn:mace:shibboleth:2.0:ad"
xmlns:pc="urn:mace:shibboleth:2.0:resolver:pc"
xmlns:resolver="urn:mace:shibboleth:2.0:resolver">
<!-- Attribute Definitions -->
<!-- The EPPN is the "standard" federated username in higher ed. For guidelines on the implementation of this attribute, refer to the Shibboleth and
eduPerson documentation. Above all, do not expose a value for this attribute without considering the long term implications. -->
- <resolver:AttributeDefinition sourceAttributeID="eduPersonPrincipalName" xsi:type="ad:Prescoped" id="eduPersonPrincipalName">
  <resolver:Dependency ref="myLDAP"/>
  <resolver:AttributeEncoder xsi:type="enc:SAML1ScopedString" encodeType="false" name="urn:mace:dir:attribute-
def:eduPersonPrincipalName"/>
  <resolver:AttributeEncoder xsi:type="enc:SAML2ScopedString" encodeType="false" name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
friendlyName="eduPersonPrincipalName"/>
</resolver:AttributeDefinition>
<!-- The uid is the closest thing to a "standard" LDAP attribute representing a local username, but you should generally "never" expose uid to federated
services, as it is rarely globally unique. -->
- <resolver:AttributeDefinition sourceAttributeID="uid" xsi:type="ad:Simple" id="uid">
  <resolver:Dependency ref="myLDAP"/>
  <resolver:AttributeEncoder xsi:type="enc:SAML1String" encodeType="false" name="urn:mace:dir:attribute-def:uid"/>
  <resolver:AttributeEncoder xsi:type="enc:SAML2String" encodeType="false" name="urn:oid:0.9.2342.19200300.100.1.1" friendlyName="uid"/>
</resolver:AttributeDefinition>
<!-- In the rest of the world, the email address is the standard identifier, despite the problems with that practice. Consider making the EPPN value the
same as your official email addresses whenever possible. -->
- <resolver:AttributeDefinition sourceAttributeID="mail" xsi:type="ad:Simple" id="mail">
  <resolver:AttributeEncoder xsi:type="enc:SAML1String" encodeType="false" name="urn:mace:dir:attribute-def:mail"/>
  <resolver:AttributeEncoder xsi:type="enc:SAML2String" encodeType="false" name="urn:oid:0.9.2342.19200300.100.1.3" friendlyName="mail"/>
</resolver:AttributeDefinition>
<!-- Data Connectors -->
- <resolver:DataConnector xsi:type="dc:LDAPDirectory" id="myLDAP" useStartTLS="{idp.attribute.resolver.LDAP.useStartTLS}"
principalCredential="{idp.attribute.resolver.LDAP.bindDNcredential}" principal="{idp.attribute.resolver.LDAP.bindDN}" baseDN="{
{idp.attribute.resolver.LDAP.baseDN}" ldapURL="{idp.attribute.resolver.LDAP.ldapURL}">
  <dc:FilterTemplate>
    <![CDATA[ %{idp.attribute.resolver.LDAP.searchFilter} ]]>
  </dc:FilterTemplate>
  <dc:ReturnAttributes>% {idp.attribute.resolver.LDAP.returnAttributes}</dc:ReturnAttributes>
  - <dc:StartTLSTrustCredential id="LDAPtoIDPCredential">
    <sec:Certificate>% {idp.attribute.resolver.LDAP.trustCertificates}</sec:Certificate>
  </dc:StartTLSTrustCredential>
</resolver:DataConnector>
</resolver:AttributeResolver>
```

Specifying the EFT SP Metadata

EFT Metadata.xml File

- Pay special attention to the entityID value.
- When specifying the entity location property be sure to include the port number if EFT is listening on any port other than the default 443.
- For this example IDP configuration set the properties WantAssertionsSigned and AuthnRequestsSigned = true.
- Place this file in the shibboleth\idp\metadata directory.

```
<?xml version="1.0"?>
- <md:EntityDescriptor entityID="https://..." xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata">
  - <md:SPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol"
    WantAssertionsSigned="true" AuthnRequestsSigned="true">
    - <md:KeyDescriptor use="signing">
      - <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        - <ds:X509Data>
          <ds:X509Certificate>MIIDtDCCApygAwIBAgIBADANBgkqhkiG9w0BAQUFADCBizELMAkGA1UEBhMCMVV
          </ds:X509Data>
        </ds:KeyInfo>
      </md:KeyDescriptor>
    - <md:KeyDescriptor use="encryption">
      - <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        - <ds:X509Data>
          <ds:X509Certificate>MIIDtDCCApygAwIBAgIBADANBgkqhkiG9w0BAQUFADCBizELMAkGA1UEBhMCMVV
          </ds:X509Data>
        </ds:KeyInfo>
      </md:KeyDescriptor>
    <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</md:NameIDFormat>
    <md:AssertionConsumerService index="1" Location="https://.../33/sp/samlv2/sso"
      Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"/>
    </md:SPSSODescriptor>
  </md:EntityDescriptor>
```

metadata-providers.xml

The metadata-providers.xml file will need to be modified to show the location of the EFT Metadata file. The EFT_Metadata.xml file referenced in the example below is created and placed in the ...shibboleth\metatdata file location. The EFT_metadata file is how we communicate that EFT is an authorized Service Provider that will be using the Shibboleth IDP.

Installing and configuring Shibboleth as the backend IDP server for use with EFT SSO

EFT Configuration

- The EFT site can just be a GS auth site with user names that exist on the LDAP backend.
- Configure EFT as follows, the idp properties can be found in the idp.properties as specified above.
- For the public key, copy the idp-signing.crt file from your shibboleth server to your EFT system and reference it in the SSO Settings. The idp-signing.crt file is automatically generated upon installation of the Shibboleth IDP server. It is located in the c:\program files(x86)\Shibboleth\idp\credentials folder.

Web SSO SAML Configuration

Service Provider:


Entity ID:

Reserved Path:

Identity Provider:

Entity ID:

POST URL:

Public Key: 

Username:

Location in assertion: NameID Attribute

Attribute name:

Identifier format:

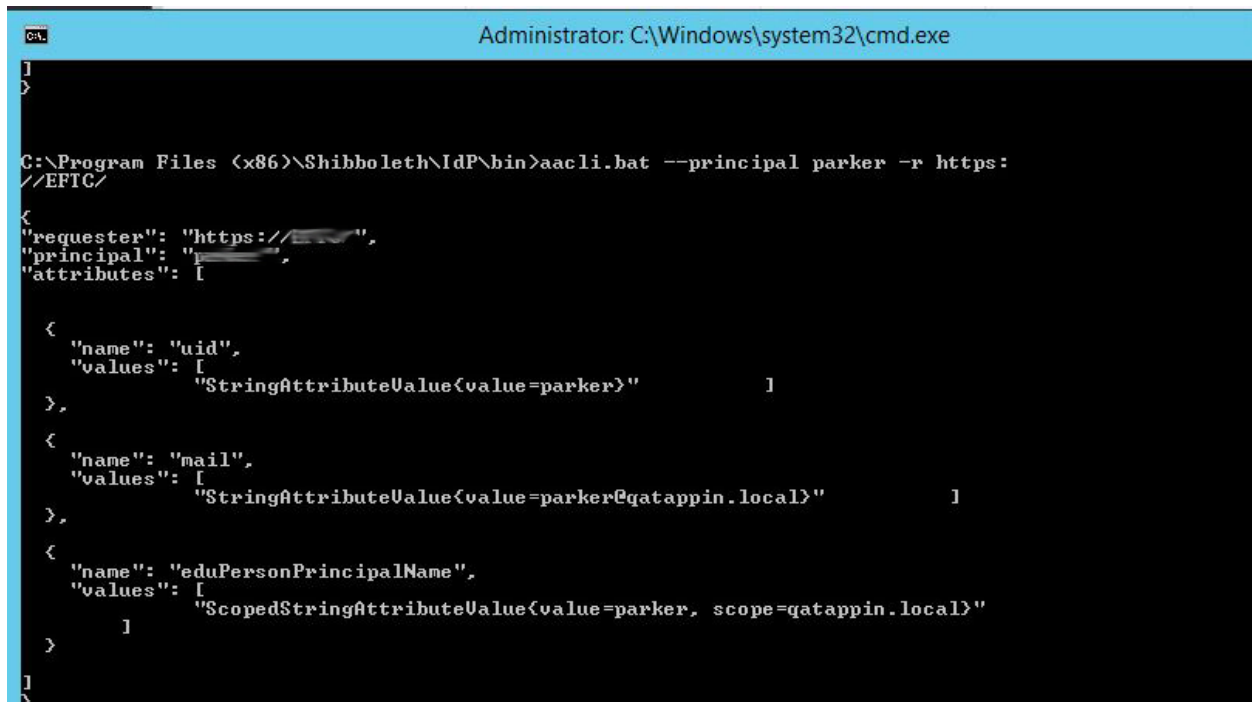
Parse the username using the regular expression:

Turn on Trace for SAML logger in logging.cfg

Installing and configuring Shibboleth as the backend IDP server for use with EFT SSO

Test configuration of release attributes

- Shibboleth has a tool called `aacli.bat` where you can test if you configured your release attributes correctly. If you do not get the expected results then you need to examine your `attribute-filter` and `attribute-resolver` files.



```
Administrator: C:\Windows\system32\cmd.exe
]
>
C:\Program Files (x86)\Shibboleth\IDP\bin>aacli.bat --principal parker -r https://EFTC/
{
  "requester": "https://[redacted]",
  "principal": "[redacted]",
  "attributes": [
    {
      "name": "uid",
      "values": [
        "StringAttributeValue<value=parker>"
      ]
    },
    {
      "name": "mail",
      "values": [
        "StringAttributeValue<value=parker@gatappin.local>"
      ]
    },
    {
      "name": "eduPersonPrincipalName",
      "values": [
        "ScopedStringAttributeValue<value=parker, scope=gatappin.local>"
      ]
    }
  ]
}
```

Shibboleth Logging

- To up the logging level can be changed in the `logback.xml` file located in the `..\shibboleth\idp\conf` directory.
- Logs are located in the `..\shibboleth\idp\logs` directory.

GlobalSCAPE Knowledge Base

<https://kb.globalscape.com/Knowledgebase/11322/Installing-and-configuring-S...>