SAML FOR BANNER

I refer to “build.banner” quite often in this document. This is simply a machine configured as an app server in ESM. Also, I talk about copying the xml for the service provider from an existing application. I built the original service provider xml from the sample files in the documentation:

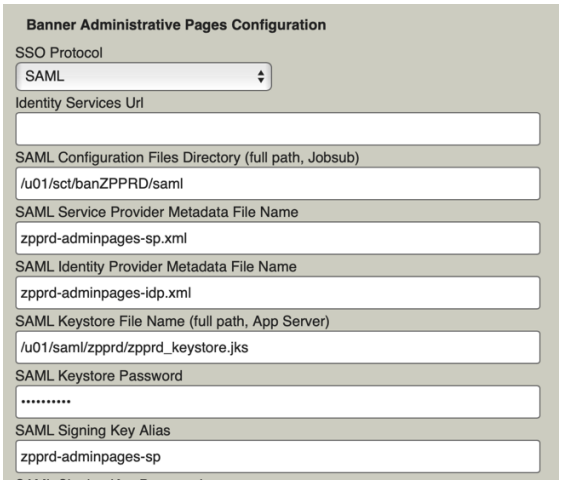
https://resources.elluciancloud.com/bundle/banner_general_acn_configure/page/c_sample_xml_files.html

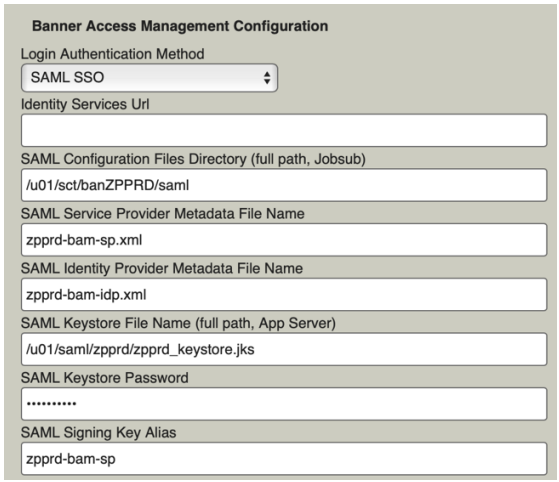
The X509Certificate field that is blank in that example XML file would need to be populated with the certificate that you generate for your Java keystore. I refer to it in this document as “mykey.”

Getting Banner Admin and Banner Access Management to work with SAML took a lot longer. I don’t talk about that in this document. I followed the instructions from Ellucian at this link to work through that:

https://resources.elluciancloud.com/bundle/banner_general_acn_configure/page/c_config_bnr_admin_app_saml2_sso.html

The key with Banner Admin and Banner Access Management was to get the settings right in ESM on the Env Settings page. It’s really a matter of getting your keystore and XML files in place on your app server and on your job submission server.





Create New Java Keystore

Use keytool to generate a new keystore
keytool -genkey -keyalg RSA -alias mykey -keystore zdevl_keystore.jks -storepass 'password' -validity 7200 -keysize 2048

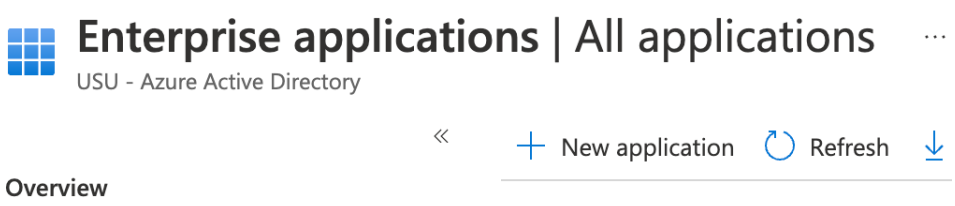
Export the certificate to PEM format
keytool -exportcert -alias mykey -keystore zdevl_keystore.jks -rfc -file mykey.pem

You can now use this keystore and the certificate in the XML sp files for each Banner application

Azure Portal Application

Log in to portal.azure.com

Go to Enterprise applications
Create a new application



Create your own application
Name it USU-(INSTANCE)-(SHORTNAME)

- i.e. USU-ZPPRD-RegistrationSS

Go to Properties and change assignment required to "No"

Go to Single sign-on
On Basic SAML configuration click Edit
Banner applications will need both the Identifier(entity ID) and the reply URL(assertion consumer service)

The identifier should be the instance name (database environment) followed by a short name for the application and then followed by '-sp'. This has to match the metadata on the Banner side.

- i.e. zpprd-registrationss-sp

The reply URL is taken from the Banner metadata xml (i.e. zpprd-registrationss-sp.xml). It will be found toward the end of the file in the AssertionConsumerService section.
If this is not created yet, copy the xml from one of the other apps (i.e. zpprd-studentss-sp.xml) and change the entity ID and URL references to match the new application.

- i.e. URL https://ss-zpprd.banner.usu.edu/StudentRegistrationSsb/saml/SSO

Click Save
Click Edit next to Attributes & Claims
Click Add new claim
Type UDC_IDENTIFIER in the name box. Use user.onpremisesamaccountname as the source attribute.
Click Save.
Click the X in the upper right to get back to the main SAML settings.
In the SAML signing certificate section, click the download button next to the federation XML.
Take this XML file and put it on build.banner in the appropriate folder under /u01/saml. Rename the xml file to match the naming convention. This is important because it needs to match the naming convention for the environment variables in the groovy file.
Also download the base64 cert and move it to build.banner.

This is the end of the Azure configuration. Now configuration the Banner application side.

Banner Application SAML

To setup SAML for a new application, start by getting a shell on build.banner.usu.edu.

- ssh root@build.banner.usu.edu

Go to the /u01/saml folder and then to the appropriate subfolder for the environment you are configuring (i.e. zpprd)

- cd /u01/saml/zpprd

You'll need to edit the Java keystore for the environment. You can see all of the aliases in the keystore by using this command:

- keytool -list -keystore zpprd_keystore.jks

You can see more details for the certs stored for each alias by adding the -v flag.

- keytool -list -v -keystore zpprd.keystore.jks

Clone the "mykey" alias to a new alias.

- keytool -keyclone -alias"mykey"-dest "zpprd-registrationss-sp"-keystore zpprd_keystore.jks

Copy the metadata file and the cert file (from follow the Azure instructions) into the environment folder following the zpprd-registrationss-idp.xml naming convention. The sp suffix is from the Banner side and the idp suffix is from the Azure side.
Now add the base64 cert (from Azure) to the keystore.

- keytool -importcert -file USU-ZPPRD-RegistrationSS.cer -keystore zpprd_keystore.jks -alias zpprd-registrationss-idp

Verify that both new aliases are in the keystore. You should see both zpprd-registrationss-idp and zpprd-registrationss-sp listed.

- keytool -list -keystore zpprd_keystore.jks

Now the keystore should be ready. The XML files need to be prepared.
Copy from an existing application.

- cp zpprd-studentss-sp.xml zpprd-registrationss-sp.xml

Edit the file and change all of the references for the old application to the new application. Both the entityID and all of the URLs need to be changed.

- entityID="zpprd-registrationss-sp"
- <idpdisco:DiscoveryResponse xmlns:idpdisco="urn:oasis:names:tc:SAML:profiles:SSO:idp-discovery-protocol" Binding="urn:oasis:names:tc:SAML:profiles:SSO:idp-discovery-protocol"

```
Location="https://ss-
zpprd.banner.usu.edu/StudentRegistrationSsb/saml/SSO/login/auth?disco=true"/>
   <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://ss-zpprd.banner.usu.edu/StudentRegistrationSsb/saml/SingleLogout"/>
   <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="https://ss-zpprd.banner.usu.edu/StudentRegistrationSsb/saml/SingleLogout"/>
   <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="https://ss-zpprd.banner.usu.edu/StudentRegistrationSsb/saml/SingleLogout"/>
   <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress</md:NameIDFormat>
   <md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-
format:transient</md:NameIDFormat>
   <md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent</md:NameIDFormat>
   <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-
format:X509SubjectName</md:NameIDFormat>
   <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://ss-zpprd.banner.usu.edu/StudentRegistrationSsb/saml/SSO" index="0"
isDefault="true"/>
   <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
Location="https://ss-zpprd.banner.usu.edu/StudentRegistrationSsb/saml/SSO" index="1"
isDefault="false"/>
   <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:PAOS"
Location="https://ss-zpprd.banner.usu.edu/StudentRegistrationSsb/saml/SSO" index="2"
isDefault="false"/>
```

Make sure that both zpprd-registrationss-sp.xml and zpprd-registrationss-idp.xml are in the zpprd folder. You are now done on build.banner.

Quite a few changes are necessary in the application's groovy file (i.e. StudentRegistrationSsb_configuration.groovy). We use environment variables in our groovy files since we have one Tomcat instance per application. You could statically set values instead.

Replace this section:

```
   banner {
      sso {
         authenticationProvider         = 'saml' //  Valid values are: 'saml' and 'cas' for SSO to work.
'default' to be used only for zip file creation.
         authenticationAssertionAttribute = 'UDC_IDENTIFIER'
      }
   }
```

With this:

```
if(System.getenv('AUTH_METHOD') == 'saml')
{
   banner {
      sso {
         authenticationProvider         = 'saml' //  Valid values are: 'saml' and 'cas' for SSO to work. 'default'
to be used only for zip file creation.
         authenticationAssertionAttribute = 'UDC_IDENTIFIER'
      }
   }
}
if(System.getenv('AUTH_METHOD') == 'cas')
{
   banner {
      sso {
         authenticationProvider         = 'cas' //  Valid values are: 'saml' and 'cas' for SSO to work. 'default'
to be used only for zip file creation.
         authenticationAssertionAttribute = 'UDC_IDENTIFIER'
      }
   }
}
```

Replace this section:

```
grails {
   plugin {
      springsecurity {
         cas {
            active = true }
```

With this:

```
grails {
   plugin {
      springsecurity {
         cas {
            if(System.getenv('AUTH_METHOD') == 'cas') { active = true }
            if(System.getenv('AUTH_METHOD') == 'saml') { active = false }
```

One more section. Replace the commented out SAML section with this:

```
if(System.getenv('AUTH_METHOD') == 'saml')
{
   if(System.getenv('AUTH_METHOD') == 'cas') { grails.plugin.springsecurity.saml.active = false }
   if(System.getenv('AUTH_METHOD') == 'saml') { grails.plugin.springsecurity.saml.active = true }
   grails.plugin.springsecurity.auth.loginFormUrl = '/saml/login'
   grails.plugin.springsecurity.saml.afterLogoutUrl ='/logout/customLogout'
   banner.sso.authentication.saml.localLogout='true' // To disable single logout set this to true,default
'false'.
   grails.plugin.springsecurity.saml.keyManager.storeFile = 'file:/usr/local/tomcat/webapps/' +
(System.getenv('APP_LONG_NAME') ?: 'StudentSelfService') + '/saml/' + (System.getenv('BANNERDB') ?:
'host') + '/' + (System.getenv('BANNERDB') ?: 'host') + '_keystore.jks'  // for unix File based Example:-
'file:/home/u02/samlkeystore.jks'
   grails.plugin.springsecurity.saml.keyManager.storePass = (System.getenv('KEYSTORE_PASSWORD') ?:
'CHANGE_ME')
   grails.plugin.springsecurity.saml.keyManager.passwords = [ ((System.getenv('BANNERDB')) + '-' +
(System.getenv('APP_SHORT_NAME')) + '-sp'): ((System.getenv('KEYSTORE_PASSWORD'))) ]  // banner-
<short-appName>-sp is the value set in Ellucian Ethos Identity Service provider setup
   grails.plugin.springsecurity.saml.keyManager.defaultKey = (System.getenv('BANNERDB') ?: 'host') + '-'
+ (System.getenv('APP_SHORT_NAME') ?: 'studentss') + '-sp'               // banner-<short-appName>-sp is
the value set in Ellucian Ethos Identity Service provider setup
   grails.plugin.springsecurity.saml.metadata.sp.file = '/usr/local/tomcat/webapps/' +
(System.getenv('APP_LONG_NAME') ?: 'StudentSelfService') + '/saml/' + (System.getenv('BANNERDB') ?:
'host') + '/' + (System.getenv('BANNERDB') ?: 'host') + '-' + (System.getenv('APP_SHORT_NAME') ?:
'studentss') + '-sp.xml'     // for unix file based Example:-'/home/u02/sp-local.xml'
   grails.plugin.springsecurity.saml.metadata.providers = [adfs: '/usr/local/tomcat/webapps/' +
(System.getenv('APP_LONG_NAME') ?: 'StudentSelfService') + '/saml/' + (System.getenv('BANNERDB') ?:
'host') + '/' + (System.getenv('BANNERDB') ?: 'host') + '-' + (System.getenv('APP_SHORT_NAME') ?:
'studentss') + '-idp.xml'] // for unix file based Example: '/home/u02/idp-local.xml'
   grails.plugin.springsecurity.saml.metadata.defaultIdp = 'adfs'
   grails.plugin.springsecurity.saml.metadata.sp.defaults = [
         local: true,
         alias: (System.getenv('BANNERDB') ?: 'host') + '-' + (System.getenv('APP_SHORT_NAME') ?:
'studentss') + '-sp',                        // banner-<short-appName>-sp is the value set in EIS Service
provider setup
         securityProfile: 'metaiop',
         signingKey: (System.getenv('BANNERDB') ?: 'host') + '-' + (System.getenv('APP_SHORT_NAME') ?:
'studentss') + '-sp',                      // banner-<short-appName>-sp is the value set in EIS Service
provider setup
         encryptionKey: (System.getenv('BANNERDB') ?: 'host') + '-' +
(System.getenv('APP_SHORT_NAME') ?: 'studentss') + '-sp',                    // banner-<short-appName>-
sp is the value set in EIS Service provider setup
         tlsKey: (System.getenv('BANNERDB') ?: 'host') + '-' + (System.getenv('APP_SHORT_NAME') ?:
'studentss') + '-sp',                        // banner-<short-appName>-sp is the value set in EIS Service
provider setup
         requireArtifactResolveSigned: false,
```

```
        requireLogoutRequestSigned: false,
        requireLogoutResponseSigned: false
    ]
}
```

Copy the application sp and idp xml files into /usr/local/tomcat/webapps/StudentRegistrationSsb/saml

There are 5 environment variables needed in the Tomcat environment if you used them in the groovy file:

- BANNERDB=zpprd
- APP_SHORT_NAME=registrationss
- APP_LONG_NAME=StudentRegistrationSsb
- KEYSTORE_PASSWORD=password
- AUTH_METHOD=saml (or cas for the SDL instances)

That should do it. You should be able to restart Tomcat and your application should be redirecting you to your Azure IDP to log in when you go to the application.