Contents

[Overview 1](#_Toc492643710)

[Installation 1](#_Toc492643711)

[Yellowfin SAML Service Provider (SP) Configuration 2](#_Toc492643712)

[SAML Identity Provider (IDP) Configuration 3](#_Toc492643713)

[Active Directory Federation Services (AD FS) 3](#_Toc492643714)

[AD FS Public Key 3](#_Toc492643715)

[Registering Yellowfin SAML Bridge Identity Provider in AD FS 4](#_Toc492643716)

[Claim Rules 5](#_Toc492643717)

[SSO service (IdpInitiatedSignOnPage) 6](#_Toc492643718)

[Yellowfin SAML Bridge Settings and User Provision 7](#_Toc492643719)

[Troubleshooting 8](#_Toc492643720)

[Signature validation failed 8](#_Toc492643721)

[Illegal Key Size 8](#_Toc492643722)

[Name ID 9](#_Toc492643723)

[COULD\_NOT\_FIND\_PERSON 9](#_Toc492643724)

# Overview

The Yellowfin SAML Bridge is a Java web application that allows for interfacing between a SAML Identity Provider, and Yellowfin. This allows for a user to use the same credentials that they use for other applications at their organization. The Yellowfin SAML Bridge in this case is a SAML Service Provider (SP). The SAML Bridge uses Yellowfin’s web services to SSO the user into Yellowfin. There is also an option for auto provisioning users the first time that they connect to Yellowfin using the SAML Bridge.

# Installation

The Yellowfin SAML Bridge is a separate Java web application that can be run within the same Tomcat instance as Yellowfin. The bridge can be installed by unzipping the YellowfinSAMLBridge.zip file into the Yellowfin/appserver/webapps/ directory.

The Yellowfin SAML Bridge uses the Yellowfin Webservice Java Library. The library (yfws.jar) that corresponds to the Yellowfin instance version should be included in the /WEB-INF/lib directory of the Yellowfin SAML Bridge.

***Note****. You can find the yfws-xxx.jar file in Yellowfin installation folder under development/lib. If your Yellowfin was upgraded, contact Yellowfin support team (**support@yellowfin.bi**) to get corresponding webservices library as Yellowfin upgrade does not upgrade development folder. Another way to get proper yfws.jar is to perform fresh Yellowfin installation of the upgraded version and copy the file from its development/lib folder.*

# Yellowfin SAML Service Provider (SP) Configuration

The Yellowfin SAML Bridge uses the OneLogin Java API to interface with SAML Identity Providers (IDP). The configuration for the SAML SP is done within the WEB-INF/classes/**onelogin.saml.properties** file.

The following properties need to be set to configure the Service Provider (The Yellowfin SAML Bridge). There are inline comments with the properties file that give more information about each option.

|  |
| --- |
| **Scenario**. You access Yellowfin via <http://yellowfin:8080>. You have SAML Bridge being installed in yellowfin/appserver/webapps/samlbridge folder. Your AD FS has **adfs.local** name. |

**onelogin.saml2.sp.entityid** - the entityId of the SAML Bridge SP. This will be the metadata URL for SAML Bridge. The URL is of the form: <scheme>://<host>:<port>/<context>/metadata.jsp. metadata.jsp is located under ‘samlbridge’ folder. This can be used to register SAML Bridge SP in AD FS.

For instance, <http://yellowfin:8080/samlbridge/metadata.jsp>

***Note.*** *Ensure that this URL is accessible from AD FS.*

**onelogin.saml2.sp.assertion\_consumer\_service.url** is the URL that handles a successful authentication. Yellowfin does it via samlbridge/acs.jsp.

For instance, [http:// yellowfin:8080/samlbridge/acs.jsp](http://myDomainName:8080/samlbridge/acs.jsp)

***Note.*** *The SP entityid must be registered with the AD FS to allow users access to this service. How to register see Registering Yellowfin SAML Bridge Identity Provider in AD FS chapter of this guide.*

**onelogin.saml2.sp.single\_logout\_service.url** is the URL that handles a logoff. samlbridge/sls.jsp file handles this.

For instance, [http:// yellowfin:8080/samlbridge/sls.jsp](http://myDomainName:8080/samlbridge/sls.jsp)

**onelogin.saml2.sp.x509cert** is the text representation of a security certificate. A self-signed certificate can be generated with:

openssl req -newkey rsa:2048 -new -x509 -days 3652 -nodes -out sp.crt -keyout sp.pem

The text representation of the sp.crt from the above command is required for this option.

**onelogin.saml2.sp.privatekey** is the text representation of the certificates private key. This is the text representation of the sp.pem file that was created by the self-signed certificate process above.

**onelogin.saml2.sp.nameidformat** is required by OneLogin SAML, should correspond to AD FS Name ID format. Can be one of:

NAMEID\_EMAIL\_ADDRESS = 'urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress';

NAMEID\_X509\_SUBJECT\_NAME = 'urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName';

NAMEID\_WINDOWS\_DOMAIN\_QUALIFIED\_NAME = 'urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName';

NAMEID\_UNSPECIFIED = 'urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified';

NAMEID\_KERBEROS = 'urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos';

NAMEID\_ENTITY = 'urn:oasis:names:tc:SAML:2.0:nameid-format:entity';

NAMEID\_TRANSIENT = 'urn:oasis:names:tc:SAML:2.0:nameid-format:transient';

NAMEID\_PERSISTENT = 'urn:oasis:names:tc:SAML:2.0:nameid-format:persistent';

NAMEID\_ENCRYPTED = 'urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted';

***Note.*** *Any changes made to the onelogin.saml.properties file will require the Yellowfin SAML Bridge to be restarted for new settings to take effect.*

# SAML Identity Provider (IDP) Configuration

The Yellowfin SAML Bridge uses the OneLogin Java API to interface with SAML Identity Providers (IDP). The configuration for the SAML IDP is also done within the WEB-INF/classes/onelogin.saml.properties file.

Each SAML Identity Provider will require different options to be filled out in the properties file. Below is listed what AD FS requires.

**onelogin.saml2.idp.entityid** = <https://adfs.local/adfs/ls/IdpInitiatedSignon.aspx?loginToRp=Yellowfin>

***Note.*** *You can find more details in ‘SSO service (IdpInitiatedSignOnPage)’ chapter of this guide.*

**onelogin.saml2.idp.single\_sign\_on\_service.url** = <https://adfs.local/adfs/ls/IdpInitiatedSignon.aspx?loginToRp=Yellowfin>

***Note.*** *Still filled in, however, maybe not required.*

**onelogin.saml2.idp.single\_logout\_service.url** = https://adfs.local/adfs/ls?wa=wsignout1.0

**onelogin.saml2.idp.x509cert** is required to sigh SAML requests before sending them to AD FS. You can find more details in ‘AD FS Public Key’ chapter of this guide.

***Note.*** *There may be issues with key size. See Troubleshooting – Illegal Key Size chapter.*

*Any changes made to the onelogin.saml.properties file will require the Yellowfin SAML Bridge to be restarted for new settings to take effect.*

# Active Directory Federation Services (AD FS)

### AD FS Public Key

You need to get valid public keyfrom ADFS (.cer file) to sing SAML requests coming from Yellowfin. This (in a text form) goes to onelogin.saml.properties:

onelogin.saml2.idp.x509cert =MIIC2DCCAcCgAwIBAgIQfdRAAWmWko1IsimA004o3TANBgkqhki…

Download signing certificate from AD FS:



Select ‘View Certificate’, go to ‘Details’, click ‘Copy to file’. Then open the file in a text editor and copy the string to onelogin.saml2.idp.x509cert.

### Registering Yellowfin SAML Bridge Identity Provider in AD FS

To register Yellowfin SAML bridge service provider, use **samlbridge/metadata.jsp**. You need to provide it in the form of URL, for instance: <http://yellowfin:8080/samlebridge/metadata.jsp>. Ensure that you can access the URL from AD FS server. It pulls the details coming from samlbridge/WEB-INF/classes/**onelogin.saml.properties**.

***Note.*** *Each time when you modify onelogin.saml.properties, you need to update the Yellowfin Relying Party Trust metadata in AD FS.*

*More details about registering service provider in AD FS can be found via* [*https://technet.microsoft.com/en-us/library/adfs2-help-how-to-add-a-relying-party-trust(v=ws.10).aspx*](https://technet.microsoft.com/en-us/library/adfs2-help-how-to-add-a-relying-party-trust%28v%3Dws.10%29.aspx)

**Add Relying Party Trust**.

Go to ‘Trust Relationship’ in AD FS manager, click on ‘Relying Party Trust’ and choose ‘Add Relying Party Trust Wizard’.

Select ‘Import data about the relying party published online or on a local network’ radio button. Type into ‘Federation metadata address (host name or URL) the URL to Yellowfin SAML Bridge metadata.jsp file. For instance, <http://yellowfin:8080/samldridge/metadata.jsp>. This will become your service provider entity id (onelogin.saml2.sp.entityid) to fill in onelogin.saml.properties file.



On the ‘Select Data Source’ page, provide a displayed name for the service provide:



This is going to be an application name visible for a user as well as part of SSO URL in onelogin.saml.properties file:

**onelogin.saml2.idp.single\_sign\_on\_service.url** = <https://adfs.local/adfs/ls/IdpInitiatedSignon.aspx?loginToRp=Yellowfin>

On the next page, select ‘I do not want to configure multi-factor authentication settings for this relying party trust at this time’. Configuring multi-factor authentication is beyond this scope. Click ‘Next’.

Select ‘Permit all users to access this relying party’ radio button. Click ‘Next’ to the end.

Once you have registered Yellowfin SAML Bridge in AD FS, you’ll be offered to set claim rules.

### Claim Rules

***Note:*** *SAML requires Name ID as part of the AD FS response, ensure that you pass it correctly in a proper format.*

For instance, you define name id in **onelogin.saml.properties** like below:

onelogin.saml2.sp.nameidformat = urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

That means you need to pass email address as a name id from AD FS. Your claim rules should look like below.

1. Request AD attributes. Click ‘Add Rule’ and choose ‘Sent LDAP Attributes as Claims’. Provide it with the name and add all the attribute you want to pass to SAML Bridge. To do automatic user provision via SAML Bridge, you need to pass at least **email address, user name, user surname**. You need to pass a proper **user id** corresponding to yellowfin authentication method (name id or email addresses). Make sure that whatever you pass as email addresses attribute indeed keeps email addresses. It can be User-Principal-Name or E-Mail-Addresses.



***Note:*** *You may want to add more AD attributes to be able to do user provision via SAML Bridge like default user role, group memberships etc. Additional modification to SAML Bridge web.xml and acs.jsp files will be required.*

1. Transform email address into name id. Click ‘Add Rule’. Select ‘Transform an Income Claim’ this time. Select ‘E-Mail Address’ as ‘Incoming claim type’. Select ‘Name ID’ as ‘Outgoing claim type’ and ‘Email’ as ‘Outgoing name ID format’ (this should correspond to **onelogin.saml2.sp.nameidformat** of onelogin.saml.properties file).



### SSO service (IdpInitiatedSignOnPage)

AD FS 2.0 provides the IdpInitiatedSignOn.aspx page to handle SAML-based IdP-initiated single sign-on (SSO). This functionality enables a user to sign on locally to the AD FS 2.0 server using the SAML protocol or to sign on to Web SSO-compatible relying party (RP) applications like Yellowfin.

This is in the URL form and goes to onelogin.saml.properties:

onelogin.saml2.idp.entityid = https://<ADFS domain name>/adfs/ls/IdpInitiatedSignon.aspx?loginToRp=<RP>

onelogin.saml2.idp.single\_sign\_on\_service.url = https://<ADFS domain name>/adfs/ls/IdpInitiatedSignon.aspx?loginToRp=<RP>

<RP> is the displayed name which you defined during registering Yellowfin SAML Bridge service provider in AD FS.

More information about IdpInitiatedSignOn.aspx can be found here:

<https://msdn.microsoft.com/en-au/library/ee895361.aspx>

# Yellowfin SAML Bridge Settings and User Provision

Settings related to the operation of the SAML Bridge are located in the WEB-INF/web.xml file. These settings describe the location of the Yellowfin Instance and the web service credentials, and the attributes for finding and automatically provisioning Yellowfin Users.

How to access Yellowfin (URL):

<init-param>

 <param-name>YellowfinWebserviceURL</param-name>

 <param-value>http://yellowfin:8080</param-value>

 </init-param>

Yellowfin webserver user (a user who can perform webservices calls with Webservice role on):

<init-param>

 <param-name>YellowfinWebserviceUser</param-name>

 <param-value>admin@yellowfin.com.au</param-value>

</init-param>

<init-param>

 <param-name>YellowfinWebservicePassword</param-name>

 <param-value>test</param-value>

 </init-param>

Enabling/disabling user provision (set to true or false):

<init-param>

 <param-name>AutoProvision</param-name>

 <param-value>true</param-value>

</init-param>

To be able to find a user (in case of ‘email addresses’ authentication method), you need provide Yellowfin with the email attribute corresponding to AD FS Claim Descriptions:



For instance,

<init-param>

 <param-name>EmailAttribute</param-name>

 <param-value>http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress</param-value>

</init-param>

To do user provision, you need to define FirstNameAttribute, LastNameAttribute and YellowfinRole. For instance, in the example below, Yellowfin gets user name and surname from AD FS and user role is defined as ‘Consumer & Collaboration’.

<init-param>

 <param-name>FirstNameAttribute</param-name>

 <param-value>http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname</param-value>

</init-param>

<init-param>

 <param-name>LastNameAttribute</param-name>

 <param-value>http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname</param-value>

</init-param>

<init-param>

 <param-name>UsernameAttribute</param-name>

 <param-value>http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress</param-value>

 </init-param>

<init-param>

 <param-name>YellowfinRole</param-name>

 <param-value>Consumer &amp; Collaborator</param-value>

</init-param>

***Note.*** *I suspect UsernameAttribute relates to Yellowfin ‘user name’ authentication method.*

# Troubleshooting

For troubleshooting, it is better to run SSO URL provided by onelogin.saml2.idp.single\_sign\_on\_service.url of onelogin.saml.properties. Ideally, on AD FS server.

### Signature validation failed

You may see the error like

ERROR c.onelogin.saml2.authn.SamlResponse - Signature validation failed. SAML Response rejected

That means that the public key which you refer in onelogin.saml.properties is not valid:

onelogin.saml2.idp.x509cert =MIIC2DCCAcCgAwIBAgIQfdRAAWmWko1IsimA004o3TANBgkqhki…

Solution:

* Get a valid certificate from AD FS;
* modify onelogin.saml.properties (onelogin.saml2.idp.x509cert);
* restart Yellowfin;
* update Yellowfin SAML Bridge relying party metadata in AD FS.

### Illegal Key Size

You may see this in Yellowfin logs:

org.apache.xml.security.encryption.XMLEncryptionException: Illegal key size

Original Exception was java.security.InvalidKeyException: Illegal key size

Solution.

When inspecting the SAML response payload below, the data is encrypted with AES-256:

EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"

By default, Java’s key size is limited to 128-bit key due to US export laws and a few countries’ import laws.

To fix:

* Download Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files:

Java 7: <http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html>

Java 8: <http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html>

* Copy local\_policy.jar and US\_export\_policy.jar to [JAVA\_HOME]/jre/lib/security.

### Name ID

SAML requires name id as part of Identity Provider response. If you see in your browser something like



you do not pass correct name id from AD FS. Ensure that you pass correct name id from AD FS and the name id matches the format which SAML bridge expects (onelogin.saml2.sp.nameidformat of onelogin.saml.properties).

Possible formats:

NAMEID\_EMAIL\_ADDRESS = 'urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress';

NAMEID\_X509\_SUBJECT\_NAME = 'urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName';

NAMEID\_WINDOWS\_DOMAIN\_QUALIFIED\_NAME = 'urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName';

NAMEID\_UNSPECIFIED = 'urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified';

NAMEID\_KERBEROS = 'urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos';

NAMEID\_ENTITY = 'urn:oasis:names:tc:SAML:2.0:nameid-format:entity';

NAMEID\_TRANSIENT = 'urn:oasis:names:tc:SAML:2.0:nameid-format:transient';

NAMEID\_PERSISTENT = 'urn:oasis:names:tc:SAML:2.0:nameid-format:persistent';

NAMEID\_ENCRYPTED = 'urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted';

Correct Yellowfin logs regarding to SAML response:

DEBUG c.onelogin.saml2.authn.SamlResponse - SAMLResponse validated --> …

…

DEBUG c.onelogin.saml2.authn.SamlResponse - SAMLResponse has NameID --> john.smith@yellowfin.bi

DEBUG c.onelogin.saml2.authn.SamlResponse - SAMLResponse has attributes: {http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress=[john.smith@yellowfin.bi]}

DEBUG com.onelogin.saml2.SamlAuth - processResponse success --> <*very long line representing signing certificate>*

### COULD\_NOT\_FIND\_PERSON

If you see this in Yellowfin logs:

INFO (AdministrationService:remoteAdministrationCall) - WebserviceException caught: 8(COULD\_NOT\_FIND\_PERSON)

That means that you switched the user provision off and this id is not a Yellowfin user.