

Contents

Overview	1
Installation	1
Yellowfin SAML Service Provider (SP) Configuration	2
SAML Identity Provider (IDP) Configuration	3
Active Directory Federation Services (AD FS)	3
AD FS Public Key.....	3
Registering Yellowfin SAML Bridge Identity Provider in AD FS.....	4
Claim Rules.....	5
SSO service (IdpInitiatedSignOnPage).....	6
Yellowfin SAML Bridge Settings and User Provision	7
Troubleshooting.....	8
Signature validation failed	8
Illegal Key Size	8
Name ID.....	9
COULD_NOT_FIND_PERSON.....	9

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`

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`

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 sign 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.

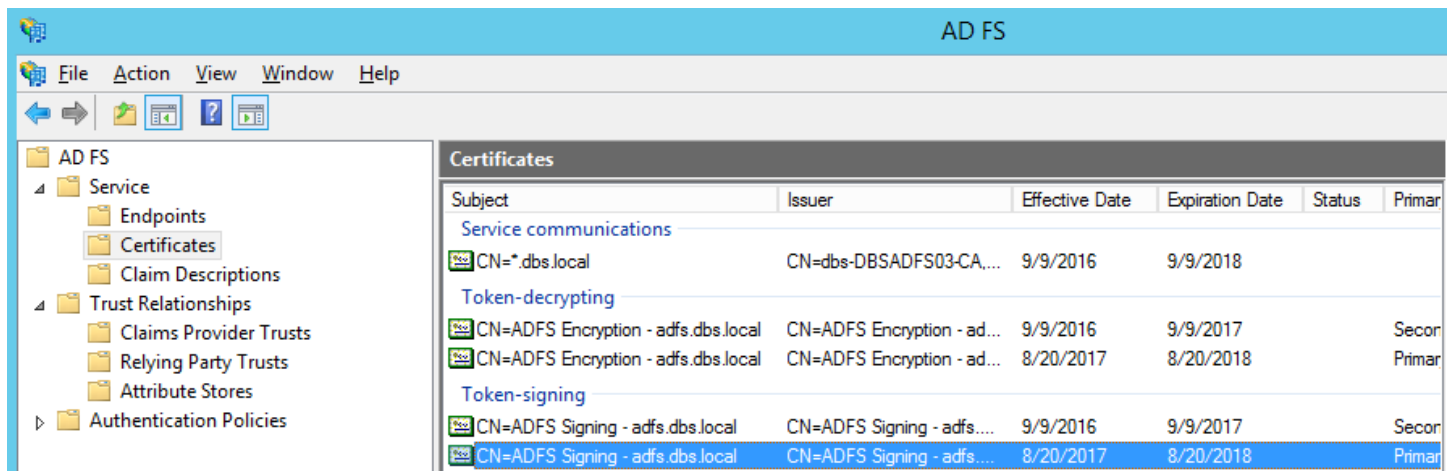
Active Directory Federation Services (AD FS)

AD FS Public Key

You need to get valid public key from ADFS (.cer file) to sign 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/samlbridge/metadata.jsp>. Ensure that you can access the URL from AD FS server. It pulls the details coming from `samlbridge/WEB-INF/classes/onenlogin.saml.properties`.

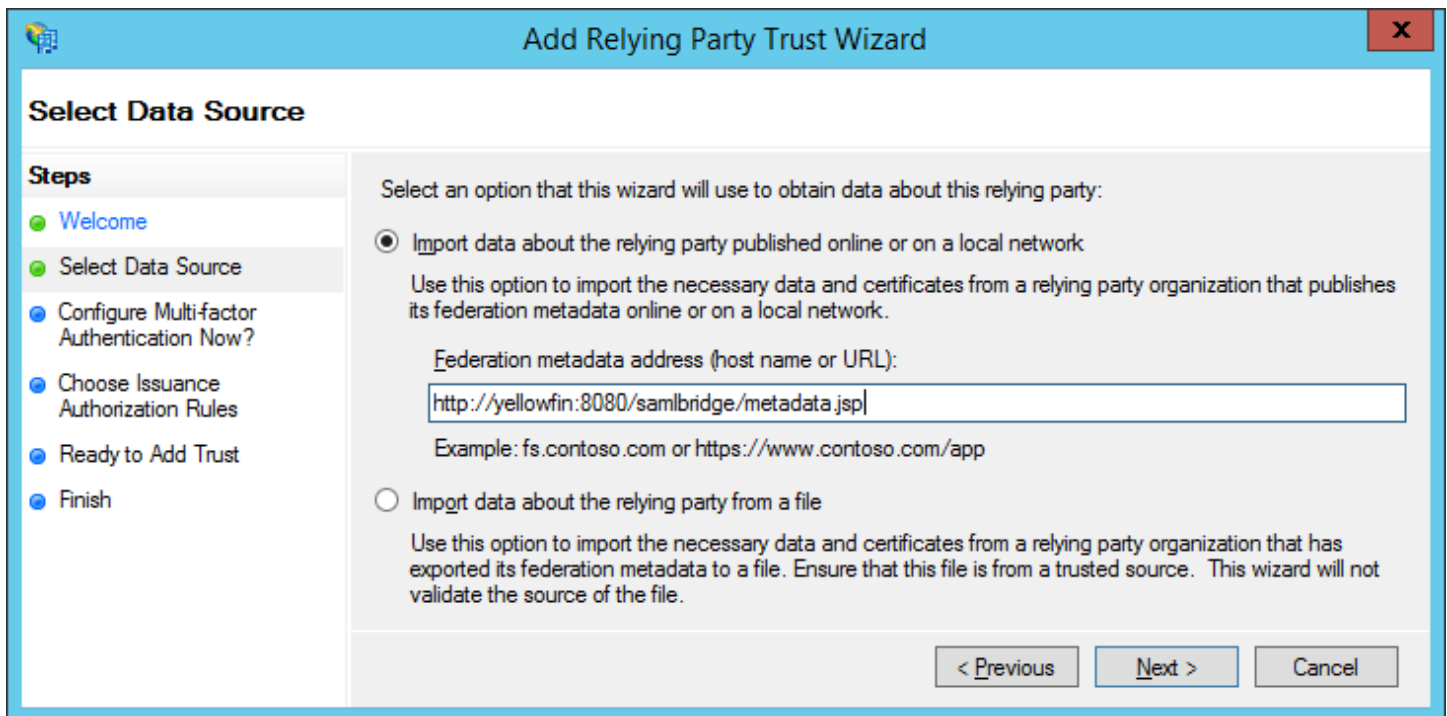
Note. Each time when you modify `onenlogin.saml.properties`, you need to update the Yellowfin Relying Party Trust metadata in AD FS.

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(v=ws.10).aspx)

Add Relying Party Trust.

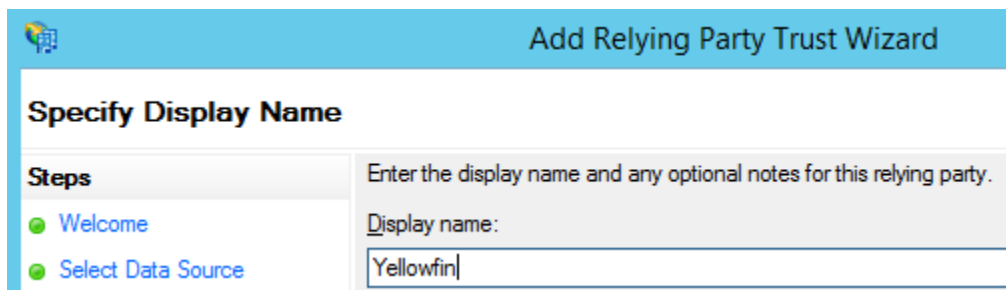
Go to 'Trust Relationship' in AD FS manager, click on 'Relying Party Trust' and choose 'Add Relying Party Trust Wizard'.

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/samlbridge/metadata.jsp>. This will become your service provider entity id (`onenlogin.saml2.sp.entityid`) to fill in `onenlogin.saml.properties` file.



The screenshot shows the 'Add Relying Party Trust Wizard' dialog box. The title bar reads 'Add Relying Party Trust Wizard'. The main heading is 'Select Data Source'. On the left, a 'Steps' pane lists: Welcome, Select Data Source (current), Configure Multi-factor Authentication Now?, Choose Issuance Authorization Rules, Ready to Add Trust, and Finish. The main area contains two radio button options. The first option, 'Import data about the relying party published online or on a local network', is selected. Below it, a text box for 'Federation metadata address (host name or URL):' contains the value 'http://yellowfin:8080/samlbridge/metadata.jsp'. An example is provided: 'Example: fs.contoso.com or https://www.contoso.com/app'. The second option, 'Import data about the relying party from a file', is unselected. At the bottom right, there are three buttons: '< Previous', 'Next >', and 'Cancel'.

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



The screenshot shows the 'Add Relying Party Trust Wizard' dialog box at the 'Specify Display Name' step. The title bar reads 'Add Relying Party Trust Wizard'. The main heading is 'Specify Display Name'. On the left, the 'Steps' pane lists: Welcome, Select Data Source (current), and Specify Display Name. The main area contains a text box for 'Enter the display name and any optional notes for this relying party.' with the label 'Display name:'. The text box contains the value 'Yellowfin'.

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 'Send 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.

Edit Rule - Email

You can configure this rule to send the values of LDAP attributes as claims. Select an attribute store from which to extract LDAP attributes. Specify how the attributes will map to the outgoing claim types that will be issued from the rule.

Claim rule name:

Rule template: Send LDAP Attributes as Claims

Attribute store:

Mapping of LDAP attributes to outgoing claim types:

	LDAP Attribute (Select or type to add more)	Outgoing Claim Type (Select or type to add more)
	User-Principal-Name	E-Mail Address
	Given-Name	Given Name
	Surname	Surname
▶	Employee-ID	uid
*		

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.

2. 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).

Edit Rule - name id

You can configure this rule to map an incoming claim type to an outgoing claim type. As an option, you can also map an incoming claim value to an outgoing claim value. Specify the incoming claim type to map to the outgoing claim type and whether the claim value should be mapped to a new claim value.

Claim rule name:

Rule template: Transform an Incoming Claim

Incoming claim type:

Incoming name ID format:

Outgoing claim type:

Outgoing name ID format:

Pass through all claim values

Replace an incoming claim value with a different outgoing claim value

Incoming claim value:

Outgoing claim value:

Replace incoming e-mail suffix claims with a new e-mail suffix

New e-mail suffix:

Example: fabrikam.com

SSO service (IdpInitiatedSignInPage)

AD FS 2.0 provides the IdpInitiatedSignIn.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 IdpInitiatedSignIn.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>
```

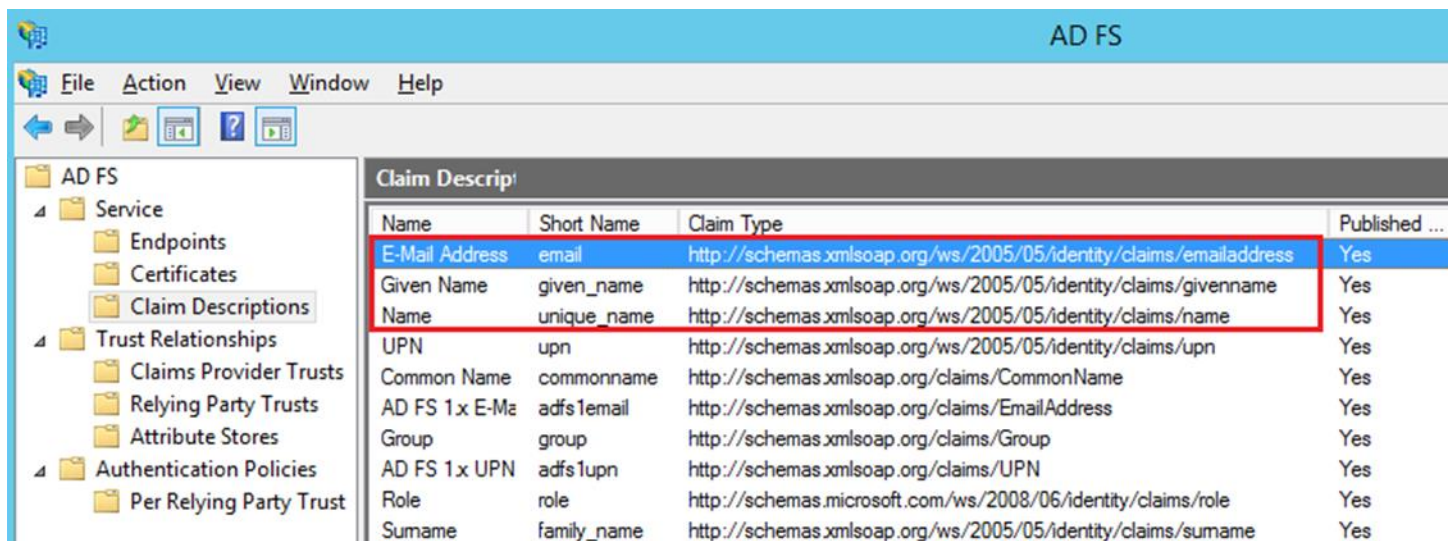
Yellowfin webservice user (a user who can perform webservicess calls with Webservice role on):

```
<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:



The screenshot shows the AD FS management console. The left pane shows the tree view with 'Claim Descriptions' selected. The right pane shows a table of claim descriptions. The 'E-Mail Address' claim is highlighted with a red box.

Name	Short Name	Claim Type	Published ...
E-Mail Address	email	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress	Yes
Given Name	given_name	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname	Yes
Name	unique_name	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name	Yes
UPN	upn	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn	Yes
Common Name	commonname	http://schemas.xmlsoap.org/claims/CommonName	Yes
AD FS 1.x E-Mail	adfs1email	http://schemas.xmlsoap.org/claims/EmailAddress	Yes
Group	group	http://schemas.xmlsoap.org/claims/Group	Yes
AD FS 1.x UPN	adfs1upn	http://schemas.xmlsoap.org/claims/UPN	Yes
Role	role	http://schemas.microsoft.com/ws/2008/06/identity/claims/role	Yes
Surname	family_name	http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname	Yes

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 & 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

https://qaportal.dbs.datacom.com.au:4452/samlbridge/acs.jsp Apache Tomcat/8.5.6 - Error... X

HTTP Status 500 - javax.servlet.ServletException: com.onelogin.saml2.exception.ValidationError: No name id found in Document.

type Exception report

message javax.servlet.ServletException: com.onelogin.saml2.exception.ValidationError: No name id found in Document.

description The server encountered an internal error that prevented it from fulfilling this request.

exception

```
org.apache.jasper.JasperException: javax.servlet.ServletException: com.onelogin.saml2.exception.ValidationError: No name id found in Document.
org.apache.jasper.servlet.JspServletWrapper.handleJspException(JspServletWrapper.java:565)
org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:466)
org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:385)
org.apache.jasper.servlet.JspServlet.service(JspServlet.java:329)
javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
```

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.