Single Sign on

Single Sign-on Overview

Yellowfin's Administration Service allows for integrating Yellowfin with essentially all third-party authentication processes. Primarily an **authentication bridge** will be used when implementing Yellowfin as a standalone application or even a tightly integrated application. But to integrate with a third-party authentication process, a **custom bridge** needs to be created. This bridge will match a user's credentials from a third-party source with those in the Yellowfin system. Usually the authentication source will provide a username, but at times a password and other user attributes, to authenticate the user.

Once a match is made with an existing Yellowfin user, the bridge will perform a Single Sign-on (**SSO**) of that user into Yellowfin. This can be done using either the **LOGINUSER** web service (which requires a password for the user to log in), or the **LOGINUSERNOPASSWORD** service, which allows the bridge to log a user in, using only their username. This seems ideal, since very rarely will there be a password available from the third-party source.

The bridge is not necessarily used to determine whether or not the user is allowed to log in. The fact that the bridge receives the username, means that the user has already been validated. However, sometimes it will be responsible for “asking” the third-party if the user is valid.

Sometimes there will be a need to auto-create the users if they do not exist in Yellowfin. This might require using additional information to create the user, like their email address, first and last names, etc. which should be sourced from the third-party application. The bridge can use the **GETUSER** or **VALIDATEUSER** web service functions to determine if a user exists in Yellowfin or not and the **ADDUSER** web service call to create a user. If bulk user creation is required, the **ADDUSERS** web service function can be called.

Part of the bridge process may also be to modify the user's Yellowfin role or group membership as part of the login process. If Yellowfin is integrated with a product where access to different content may change, it may be required to update this group membership during the login process. This would require sourcing information from the third-party source about which groups a user should be added to/removed from. The **UPDATEUSER** web service call will allow a user’s role to be modified and the **INCLUDEUSERINGROUP** or **EXCLUDEUSERFROMGROUP** calls can be used to add or remove from groups that determine which Yellowfin content they can access.

The bridge can be implemented in many ways, such as being integrated as:

- a part of Yellowfin itself (as a JSP, Servlet or Filter within the Yellowfin web application);
- a standalone application (GUI, console or web application) that communicates with the third-party source and Yellowfin;
- a part of the third-party application itself.

The best place to implement the bridge will depend on the environment and components involved.

When implementing within the Yellowfin container, the various implementation methods will allow for different functionality to be included. JSPs and Servlets allow for implementing code when the user is directed to a particular URL, whereas filters allow for checking authentication on any URL requested from the Yellowfin system.

Here is a basic process of what a Yellowfin authentication bridge needs to do:

1. Get details via cookie, file, or network connection.
2. Check if the user already exists?
3. If user doesn't exist, then create user with the details provided.
4. If required, update user's details (such as, role, group, etc.)
5. Perform a SSO call to log the user in.

Note: If your authentication provider supports SAML, the Yellowfin SAML bridge can be used to SSO users.

Single Sign On Functions

- LOGINUSER
- LOGINUSERNOPASSSWORD

LOGINUSER

This service connects to Yellowfin and retrieves a logon token for a given user. The user is specified using a user ID (such as an email address or another type of ID depending on the Logon ID method). When this token is passed with the Yellowfin Logon URL, it will disable the login screen for the authenticated users and their session will start immediately.

This function can also be used to pass different login session parameters in order to perform additional tasks, such as hide the Yellowfin header, or navigate to a specific report or dashboard directly after logging in. To learn more about these login session options, refer to this section.

Request Elements

The LOGINUSER function will Single Sign On a given user into Yellowfin. The following elements will be passed with this request:

Note: The contents of the AdministrationPerson object will be used to define the user being logged in.

Request Element	Data Type	Description
LoginId	String	Yellowfin web services admin user Id. This can be the user ID or the email address, depending on the Logon ID method. This Yellowfin account must have the “web services” role enabled, and must belong to the Default (i.e. Primary) Org.
Password	String	Password of the above account.
OrgId	Integer	Default (i.e. Primary) organization ID within Yellowfin. Always set this to 1.
Function	String	Web services function. Set this to "LOGINUSER".
Person	AdministrationPerson	The AdministrationPerson object holding details of the user that needs to log in. **Note:** See table below.
OrgRef	String	Client Org Internal Reference Id (optional). This will log the user into a given Client Org. If this is not set, then the user will be prompted with the Client Org Selection page on login.

These are the main parameters that you need to set in the **AdministrationPerson** object for this function:

AdministrationPerson Element	Data Type	Description
UserId	String	User ID of the user that you wish to log in. This can be the user ID or the email address, depending on the Logon ID method.
Password	String	Password of the new user.

The following SOAP example shows the parameters that you can pass to this call:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:web="http://webservices.web.
mi.hof.com/">
   <soapenv:Header/>
   <soapenv:Body>
      <web:remoteAdministrationCall>
         <arg0>
            <loginId>admin@yellowfin.com.au</loginId>
            <password>test</password>
            <orgId>1</orgId>
            <function>LOGINUSER</function>
            <person>
               <userId>admin@yellowfin.com.au</userId>
               <password>test</password>
            </person>
         </arg0>
      </web:remoteAdministrationCall>
   </soapenv:Body>
</soapenv:Envelope>
```

Response Elements

The response returned will contain these parameters:

Response Element	Data Type	Description
StatusCode	String	Status of the web service call. Possible values include: • SUCCESS • FAILURE
LoginSessionId	String	A unique login token. This token is appended to the Login URL to take a user directly into Yellowfin.

The service will return the below response, according to our SOAP example:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:remoteAdministrationCallResponse xmlns:ns2="http://webservices.web.mi.hof.com/">
         <return>
            <errorCode>0</errorCode>
            <loginSessionId>689bce5624f1e5a312eb5ef7801ed9fc</loginSessionId>
            <messages>Successfully Authenticated User: admin@yellowfin.com.au</messages>
            <messages>Web Service Request Complete</messages>
            <sessionId>c8021e27fcc2ce507ff17ec1846919a5</sessionId>
            <statusCode>SUCCESS</statusCode>
         </return>
      </ns2:remoteAdministrationCallResponse>
   </S:Body>
</S:Envelope>
```

Usage Instructions

See below for step-by-step instructions on how to perform this call, using a Java example:

- In our example admin@yellowfin.com.au is a web service admin user, therefore we will make the following request for this call.

```
AdministrationServiceRequest rsr = new AdministrationServiceRequest();

rsr.setLoginId("admin@yellowfin.com.au");
rsr.setPassword("test");
rsr.setOrgId(1);
rsr.setFunction("LOGINUSER");
```

- The LOGINUSER function requires the AdministrationPerson object to define a user to log in. The example below shows this:

```
AdministrationPerson ap = new AdministrationPerson();

ap.setUserId("user@yellowfin.com.au");
ap.setPassword("usertest");
rsr.setPerson(ap);
```

- If the user is a member of multiple client organizations, you can specify a particular organization to log in to. For instance:

```
rsr.setOrgRef("org1");
```

  Where "org1" refers to the client organization reference ID. If this reference ID is not provided, then the user will be redirected to the Client Org Selection page upon logging in.

- If the user account does not exist in Yellowfin, then you will receive web service error 25: COULD_NOT_AUTHENTICATE_USER.

- The LOGINUSER function also allows for different login session parameters to be specified via the setParameters() method.
  For example, the code below will log the user, user@yellowfin.com.au, into Yellowfin, but the Yellowfin header will not be displayed, and the user will be taken to the timeline page once the call is performed.

```
String[] parameters = new String[] {"ENTRY=TIMELINE","DISABLEHEADER=TRUE"};
rsr.setParameters(parameters);
```

For more options on similar login session options, click here.

- Once the request is configured, perform the call:

```
AdministrationServiceResponse rs = adminService.remoteAdministrationCall(rsr);
```

Initialize the Administration web service. Click here to learn how to do this.

- The response will contain the following parameters:

Response Element	Data Type	Description
StatusCode	String	Status of the web service call. Possible values include: ○ SUCCESS ○ FAILURE
LoginSessionId	String	A unique login token. This token is appended to the Login URL to take a user directly into Yellowfin.

Redirecting to Yellowfin with the Login Token

Using the token received from the web service call (the contents of AdministrationResponse.LoginSessionId), you can forward the user to the URL:

```
http://<YELLOWFIN-SERVER>/logon.i4?LoginWebserviceId=<TOKEN>
```

This URL will bypass the authentication screen in Yellowfin and take the user directly into Yellowfin.

> ⚠ The token has a limited validity period. It must be used within 5 minutes, and once it has been used, it cannot be used again. To make subsequent calls from a third-party application into Yellowfin, you must call the LOGINUSER web service again.

Using the Token with the JavaScript API

The SSO token can also be used with embedded JavaScript API widgets. The token is added to the scriptlet URL like this:

```
<script type="text/javascript" src="http://localhost/JsAPI?dashUUID=e9a6ab0a-bcb0-4fe6-9663-
4dd33e58f08e&token=<TOKEN>"></script>
```

Login Session Options

You can pass variables/switches that toggle functionality only for the session created via this Single Sign On request. These options can be enabled by passing them via the Parameters attribute in the AdministrationRequest, or by appending them to the redirection URL. Click here to read more about this.

Complete Usage Example

You can use the following LOGINUSER example. To try it out, follow these steps:

1. Copy the below code and save it as ws_admin_singlesignon.jsp.
2. Place this file in the root folder, that is *Yellowfin/appserver/webapps/ROOT*.
3. Adjust host, port, admin user and user to login details according to your environment.
4. Run *http://<host>:<port>/ws_admin_singlesignon.jsp* from your Internet browser.

```
<%
/*       ws_admin_singlesignon.jsp         */
%>
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ page import="com.hof.util.*, java.util.*, java.text.*" %>
<%@ page import="com.hof.web.form.*" %>
<%@ page import="com.hof.mi.web.service.*" %>
<%
String url = "http://localhost:8080";         //provide your Yellowfin URL

AdministrationServiceService s_adm = new AdministrationServiceServiceLocator("localhost",8080,"/services
/AdministrationService", false);        // adjust host and port number


AdministrationServiceSoapBindingStub adminService = (AdministrationServiceSoapBindingStub) s_adm.
getAdministrationService();
AdministrationServiceRequest rsr = new AdministrationServiceRequest();

rsr.setLoginId("admin@yellowfin.com.au");          // provide your Yellowfin webservices admin account
rsr.setPassword("test")                            // change to be the password of the account above
rsr.setOrgId(1);
rsr.setFunction("LOGINUSER");

AdministrationPerson ap = new AdministrationPerson();
ap.setUserId("user@yellowfin.com.au");                  // provide existing Yellowfin user to login
ap.setPassword("usertest");                             // password of the user above

rsr.setPerson(ap);

String[] parameters = new String[] {"ENTRY=TIMELINE","DISABLEHEADER=TRUE"};
rsr.setParameters(parameters);

AdministrationServiceResponse rs = adminService.remoteAdministrationCall(rsr);


String token = "";
if ("SUCCESS".equals(rs.getStatusCode()) ) {
               token = rs.getLoginSessionId();
               response.sendRedirect(url + "/logon.i4?LoginWebserviceId=" + token);
} else {
               out.write("Single Sign on Failure");
               return;
}
%>
```

LOGINUSERNOPASSSWORD

The LOGINUSERNOPASSWORD web service will allow to login a user using only their username.

Enabling Functionality

An extra parameter needs to be added to the Configuration table of the Yellowfin database to enable this functionality:

IpOrg	ConfigTypeCode	ConfigCode	ConfigData
1	SYSTEM	SIMPLE_AUTHENTICATION	TRUE

Note: You may need to restart Yellowfin for the database change to take effect.

You will receive a web service error 26: UNSECURE_LOGIN_NOT_ENABLED if this configuration option is not added to the Yellowfin database.

Parameter options for this call are the same as LOGINUSER, except for:

- AdministrationRequest.Function will be set to LOGINUSERNOPASSWORD
- AdministrationPerson.Password can be blank

top