Web Services

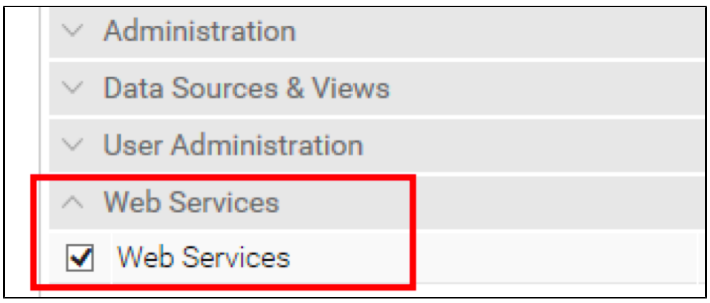
Overview

Web Services are used for managing communication between an OEM application and Yellowfin. These are XML based and independent of the programming language used to develop the OEM application.

The web service interface Yellowfin exposes use SOAP, which is a protocol widely used for web services. Yellowfin provides a Java Web Service API for connecting to the SOAP web services, but it is also possible to connect from practically any other programming language or environment such as .NET, Ruby and Python.

Enabling Web Services

You need to have a Yellowfin user with rights to perform web services calls. Ensure that this is turned on in the user role:



You will need the web services admin user's details (login name and password) to perform web service calls.

Java API

The Yellowfin Web Service API contains pre-generated stubs. This can be used directly in applications that are developed in Java, or other languages that support Java integration, such as Cold Fusion or Lotus Script. This makes integration slightly simpler as each request is not required to be manually generated, since most of the Web Services are wrapped by a standard Java function.

API can be called internally under Yellowfin Tomcat using JSP. The code samples can be found in the *yellowfin/development/examples/webservices* folder, once Yellowfin is installed. All you need to do is to copy the JSP files into the *Yellowfin/appserver/webapps/ROOT* folder and adjust the host, port number, and user details in the JSP files according to your environment. We recommend ensuring that you can achieve what you want using this method prior to replicating this with other languages or environments.

To call web services externally, you will need:

- *yfws-<date>.jar* which can be found in the *development/lib* folder in the Yellowfin directory. Note: Do not forget to get a new yfws-xxx.jar file after a Yellowfin upgrade (you need to download a corresponding yfws-xxx.jar file from the Yellowfin website).
- Apache Axis: https://axis.apache.org/axis/

A full objects' definitions can be found at Yellowfin/development/doc/webservices/Javadoc/index.html

There are two ways of calling Java API:

- Using pre-built Java functions. This is limited by functionality, however, all basic functions like performing SSO, rendering reports, and passing filters are covered.

The code samples regarding this method can be found in the *development/examples/webservices* folder. See the jsp files with 'api' in their names. A good starting point is copying files with 'mobile' in their names, into the Yellowfin ROOT folder and explore.

- Performing direct SOAP calls using Java generated stubs off Yellowfin WSDL. Expand the section below for further details.

All the code samples under Administration Service and Report Service sections are explained using SOAP calls in Java.

To initialize Administration service:

AdministrationServiceService s_adm = new AdministrationServiceServiceLocator(<host>,<port>, <ServicePath>, <ssl>) ;

AdministrationServiceSoapBindingStub rssbs_admin = (AdministrationServiceSoapBindingStub) s_adm.getAdminService();

To initialize Report service:

ReportServiceService s_rpt = new ReportServiceServiceLocator(<host>, <port>, <ServicePath>, <ssl>);

ReportServiceSoapBindingStub rssbs_report = (ReportServiceSoapBindingStub) s_rpt.getReportService();

Where

Parameter	Type	Description
host	String	Yellowfin server
port	Integer	http port number to access Yellowfin server
ServicePath	String	Path to the service Administration service path: "/services/AdministartionService" Report service path: "/services/ReportService"
ssl	Boolean	If SSL enabled: true

The primary objects include:

- **AdministrationServiceRequest:** An object that defines the type of call being made to the web service.
- **AdministrationServiceResponse:** An object returned by the web service.

There are two groups of web services:

- Administration Service allows to manage users and client orgs, and perform Single-Sign-On. These are enabled with any Yellowfin license.
- Report Service allows to load reports/dashboard definitions, render reports into your interface, etc. This requires a Server license.

Languages other than Java

When developing against Yellowfin Web Services, it is possible to generate functional stubs against the WSDL definitions. These definitions can be found at http://<yellowfin-server>:<port>/services, for instance, http://localhost:8080/services.

The functional stubs will allow developers to make standard function calls in their native programming language which will directly communicate with the Web Services provided by Yellowfin. The process of creating function stubs should also generate any objects required by the webservice.

With .NET, we recommend generating stubs from JAX web services. You should be able to hit the JAX web services at: http://<yellowfin-host>/webservices/Hello. It will display something like this:

Web Services

Endpoint	Information
Service Name: {http://webservices.web.mi.hof.com/}HelloServiceService Port Name: {http://webservices.web.mi.hof.com/}HelloServicePort	Address: http://localhost:8888/webservices/Hello WSDL: http://localhost:8888/webservices/Hello?wsdl Implementation class: com.hof.mi.web.webservices.HelloService
Service Name: {http://webservices.web.mi.hof.com/}LegacyReportServiceService Port Name: {http://webservices.web.mi.hof.com/}LegacyReportServicePort	Address: http://localhost:8888/webservices/LegacyReportService WSDL: http://localhost:8888/webservices/LegacyReportService?wsdl Implementation class: com.hof.mi.web.webservices.LegacyReportService
Service Name: {http://webservices.web.mi.hof.com/}LegacyAdministrationServiceService Port Name: {http://webservices.web.mi.hof.com/}LegacyAdministrationServicePort	Address: http://localhost:8888/webservices/LegacyAdministrationService WSDL: http://localhost:8888/webservices/LegacyAdministrationService?wsdl Implementation class: com.hof.mi.web.webservices.LegacyAdministrationService
Service Name: {http://webservices.web.mi.hof.com/}LegacyVersionServiceService Port Name: {http://webservices.web.mi.hof.com/}LegacyVersionServicePort	Address: http://localhost:8888/webservices/LegacyVersionService WSDL: http://localhost:8888/webservices/LegacyVersionService?wsdl Implementation class: com.hof.mi.web.webservices.LegacyVersionService

Connect your clients to the listed WSDL URLs.

There may be issues where data types between Java and .Net are not compatible. For example, Integer types that send through zero, rather than null. You might need to manually change the References.cs file to update the datatypes.

You can use Axis generated WSDL (http://<yellowfin-server>:<port>/services) with PHP. See examples of performing SSO using PHP.

> *See Administration Service or Report Service for more information.*

top