# Calling Yellowfin 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.

You need to have a Yellowfin user with the correct role functionality to perform web services calls.

## Calling 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.

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 these 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.

You can still call web services **externally**, that is outside of Yellowfin Tomcat. To do so, you will need:

- ***yfws-<date>.jar*** which can be found in the *development/lib* folder in the Yellowfin directory.

  

  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**: Refer to https://axis.apache.org/axis/ for more information on this.

A full object definition can be found at *Yellowfin/development/doc/webservices/Javadoc/index.html*

There are two ways of calling the Yellowfin Web Service API: via pre-built Java functions or by performing SOAP calls. Choosing a method is largely dependant on your application environment. If you have a Java environment, the recommended method would be to use pre-built Java functions, otherwise you can perform SOAP calls manually.

### Using Pre-built Java Functions

You can use pre-built Java functions to call Yellowfin API. This makes development a little bit easier as you are using pre-built functions, rather than configuring each request manually.

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 exploring.

### Performing SOAP Calls

You can perform direct SOAP calls using Java generated stubs off Yellowfin WSDL.

All the code samples in the Administration Service and Report Service sections are explained using SOAP calls in Java. All the web service examples included here are explained assuming that you will call Yellowfin API from a Yellowfin Tomcat server (that means you use JSP and all your files go to *Yellowfin/appserver/webapps/ROOT* folder). Using languages other than Java will not bring much complexity to the code.

### Initializing Administration Web Services

Use this command to initialize the Administration web services:

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

AdministrationServiceSoapBindingStub adminService = (AdministrationServiceSoapBindingStub) s_adm.
getAdministrationService();
```

### Initializing Report Web Services

Use this command to initialize the Report web services:

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

ReportServiceSoapBindingStub reportService = (ReportServiceSoapBindingStub) s_rpt.getReportService();
```

The primary objects included in these parameters are (covered in detail in the Administration Object Definition section):

- AdministrationServiceRequest
- AdministrationServiceResponse

### Storing Yellowfin Session IDs

Every web service response retrieves a Yellowfin session ID. Each time a call is made without specifying a session ID, Yellowfin opens a new session. This is not suitable for some cases (for instance, if trying to pass dashboard filters to dashboard reports, all the reports must be called within the same Yellowfin session) as there may be a memory issue with too many sessions being opened. To overcome this problem, you could store the response parameter, sessionId, and pass it to the next calls:

```
String savedSessionID = ssr.getSessionId();

...

AdministrationServiceRequest sr = new AdministrationServiceRequest();
sr.setSessionId(savedSessionID);
```

### Code Samples for Administration Services

Assuming you have Yellowfin running on http port 8080 with SSL disabled, see the following example to initialize an Administration service:

```
AdministrationServiceService s_adm = new AdministrationServiceServiceLocator("localhost",8080,"/services
/AdministrationService", false);

AdministrationServiceSoapBindingStub adminService = (AdministrationServiceSoapBindingStub) s_adm.
getAdministrationService();
```

Once you configure the request, you can call Yellowfin using the remoteAdministrationCall() function of the AdministrationServiceSoapBindingStub object:

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

### Code Samples for Report Services

Assuming you have Yellowfin running on 8080 http port with SSL disabled, see the following example to initialize a Report service:

```
ReportServiceService s_rpt = new ReportServiceServiceLocator("localhost",8080,"/services/ReportService",
false);

ReportServiceSoapBindingStub reportService = (ReportServiceSoapBindingStub) s_rpt.getReportService();
```

Once you configure the request, you can call Yellowfin using the remoteReportCall() function of the ReportServiceSoapBindingStub object:

```
ReportServiceResponse rs = reportService.remoteReportCall(rsr);
```

## Other Languages

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 web service.

### Microsoft .Net Integration

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 a page with WSDL URLs:

**Web Services**

| Endpoint | Information |
|---|---|
| Service Name: {http://webservices.web.mi.hof.com/}HelloServiceService<br>Port Name: {http://webservices.web.mi.hof.com/}HelloServicePort | Address: http://localhost:8083/webservices/Hello<br>WSDL: http://localhost:8083/webservices/Hello?wsdl<br>Implementation class: com.hof.mi.web.webservices.HelloService |
| Service Name: {http://webservices.web.mi.hof.com/}LegacyReportServiceService<br>Port Name: {http://webservices.web.mi.hof.com/}LegacyReportServicePort | Address: http://localhost:8083/webservices/LegacyReportService<br>WSDL: http://localhost:8083/webservices/LegacyReportService?wsdl<br>Implementation class: com.hof.mi.web.webservices.LegacyReportService |
| Service Name: {http://webservices.web.mi.hof.com/}LegacyAdministrationServiceService<br>Port Name: {http://webservices.web.mi.hof.com/}LegacyAdministrationServicePort | Address: http://localhost:8083/webservices/LegacyAdministrationService<br>WSDL: http://localhost:8083/webservices/LegacyAdministrationService?wsdl<br>Implementation class: com.hof.mi.web.webservices.LegacyAdministrationService |
| Service Name: {http://webservices.web.mi.hof.com/}LegacyVersionServiceService<br>Port Name: {http://webservices.web.mi.hof.com/}LegacyVersionServicePort | Address: http://localhost:8083/webservices/LegacyVersionService<br>WSDL: http://localhost:8083/webservices/LegacyVersionService?wsdl<br>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.

### PHP

You can use Axis generated WSDL (http://<yellowfin-server>:<port>/services) with PHP.

**Previous topic:** Enable web services