Schedule Management Services

The data structures of Yellowfin's schedule management web services consist of two main objects (which are abstract types): **AdministrationSchedule** and **ScheduleFrequency**. AdministrationSchedule describes common properties of all schedules, including a code for determining the object subtype and most importantly containing a ScheduleFrequency object. This ScheduleFrequency object holds common data for all frequency types and is the primary way of editing schedules. Both AdministrationSchedule and ScheduleFrequency have several concrete classes representing each type of schedule and frequency available in Yellowfin. These concrete classes contain user-friendly methods to access and manipulate type-specific schedule and frequency data with client-side validation, but this is only the case when using the Java bindings.

Most attributes of an AdministrationSchedule are not editable, except for the isActive attribute. Note that this only means that the attributes are ignored by the service backend, the beans themselves are still able to be edited.

This web service loads all available schedules in the current organization that is specified with the OrgId parameter.

Request Parameters

The following parameters should be passed with this request:

Request Element	Data Type	Description
LoginId	String	An admin account to connect to Yellowfin web services. This can be the user ID or the email address, depending on the Logon ID method. This 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 service function. Set this to "LISTSCHEDULES".

Request Example

Below is a SOAP XML example for this request:

```
<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>LISTSCHEDULES</function>
         </arg0>
      </web:remoteAdministrationCall>
   </soapenv:Body>
</soapenv:Envelope>
```

Response Parameters

The returned response will contain these parameters:

Response Element	Data Type	Description

StatusCode	String	Status of the web service call. Possible values include: • SUCCESS • FAILURE
Schedules	AdministrationSchedule[]	This will contain an array of AdministrationSchedule objects describing all the available schedules.

Response Example

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>
            <messages>Successfully Authenticated User: admin@yellowfin.com.au</messages>
            <messages>Web Service Request Complete</messages>
            <schedules xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
            <schedules>
               <frequency>
                  <frequencyCode>MONDAY</frequencyCode>
                  <frequencyUnit>1</frequencyUnit>
                  <localRunTime>0</localRunTime>
                  <localTimezoneCode>AUSTRALIA/SYDNEY</localTimezoneCode>
               </frequency>
               <lastRunDateTimeGMT>2018-02-25T00:00:00+11:00</lastRunDateTimeGMT>
               <lastRunStatus>SUCCESS</lastRunStatus>
               <nextRunDateTimeGMT>2018-03-12T00:00:00+11:00</nextRunDateTimeGMT>
               <scheduleActive>true</scheduleActive>
               <scheduleDescription>Athlete</scheduleDescription>
               <scheduleUUID>75a2f5b5-162b-49b5-b197-53643f7dc0de</scheduleUUID>
            </schedules>
            <schedules xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
            <schedules>
               <frequency>
                  <frequencyCode>SATURDAY</frequencyCode>
                  <frequencyUnit>6</frequencyUnit>
                  <localRunTime>0</localRunTime>
                  <localTimezoneCode>AUSTRALIA/SYDNEY</localTimezoneCode>
               </frequency>
               <nextRunDateTimeGMT>2018-03-10T00:00:00+11:00</nextRunDateTimeGMT>
               <scheduleActive>false</scheduleActive>
               <scheduleDescription>Common Filters</scheduleDescription>
               <scheduleUUID>fa757330-b4a8-4047-9b96-745a48b1d1b7</scheduleUUID>
            </schedules>
            <schedules xsi:nil="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
            <schedules>
               <frequency>
                  <frequencyCode>SATURDAY</frequencyCode>
                  <frequencyUnit>6</frequencyUnit>
                  <localRunTime>0</localRunTime>
                  <localTimezoneCode>AUSTRALIA/SYDNEY</localTimezoneCode>
               </frequency>
               <lastRunDateTimeGMT>2018-02-19T00:00:00+11:00</lastRunDateTimeGMT>
               <lastRunError>com.hof.util.ActionErrorsException: java.lang.NullPointerException</lastRunError>
               <lastRunStatus>FAILURE</lastRunStatus>
               <nextRunDateTimeGMT>2018-03-10T00:00:00+11:00</nextRunDateTimeGMT>
               <scheduleActive>false</scheduleActive>
               <scheduleDescription>Common Filters</scheduleDescription>
               <scheduleUUID>f732c7a4-b81a-4788-8038-6771229596c1</scheduleUUID>
            </schedules>
            <sessionId>ba906c4149a72b2f3c750467a31adf72</sessionId>
            <statusCode>SUCCESS</statusCode>
         </return>
      </ns2:remoteAdministrationCallResponse>
   </S:Body>
</S:Envelope>
```


Instructions

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

- Start with a basic request for this function, which includes logging in as the admin user and specifying the web service call to perform:

```
AdministrationServiceRequest rsr = new AdministrationServiceRequest();

rsr.setLoginId(this.username);
rsr.setPassword(this.password);
// This is the primary organization
rsr.setOrgId(new Integer(1));

rsr.setFunction("LISTSCHEDULES");
```

- You may even identify a specific client organization:

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

- Once the request is configured, perform the call:

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

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

- The response will contain the following parameters: StatusCode and Schedules. (See details in the Response Parameters table above.)

Complete Example

Below is a full example of this web service call. To use it for yourself, carry out the following the steps:

1. Copy the code and save it as ws_ listschedules.jsp.
2. Put the file in the root folder: *Yellowfin/appserver/webapps/ROOT*.
3. Adjust the host, port, and admin user according to your environment.
4. Run *http://<host>:<port>/ws_ listschedules.jsp* from your Internet browser.

```
/*
 * LISTSCHEDULES Example. ws_listschedules.jsp
 * A more complete example can be found in ws_admin_schedule_management.jsp
 */

<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ page import="java.text.*" %>
<%@ page import="java.util.*" %>
<%@ page import="com.hof.mi.web.service.*" %>
<%@ page import="com.hof.mi.web.service.schedule.*" %>
<%@ page import="com.hof.web.form.*" %>

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 web services admin account
rsr.setPassword("test");                           // change to the password of the above account
rsr.setOrgId(1);
rsr.setFunction("LISTSCHEDULES");

AdministrationServiceResponse rs = adminService.remoteAdministrationCall(rsr);

if ("SUCCESS".equals(rs.getStatusCode()) ) {
        AdministrationSchedule[] schedules = rs.getSchedules();
        out.write("Loaded " + schedules.length + " schedules: <br>");

        for (AdministrationSchedule as: schedules) {
                out.write("Schedule " + as.getScheduleUUID() + "<br>");
        }

} else {
    out.write("Failure");
    out.write(" Code: " + rs.getErrorCode() );
}
```

This web service is used to load a specified schedule.

Request Parameters

The following parameters should be passed with this request:

Request Element	Data Type	Description
LoginId	String	An admin account to connect to Yellowfin web services. This can be the user ID or the email address, depending on the Logon ID method. This 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 service function. Set this to "LOADSCHEDULE".
Parameters	String[]	The UUID of the schedule to be loaded. This should be set as the first element in the passed parameters array. This must be an existing UUID.

Response Parameters

The returned response will contain these parameters:

Response Element	Data Type	Description
StatusCode	String	Status of the web service call. Possible values include: • SUCCESS • FAILURE
Schedules	AdministrationSchedule[]	This will contain an array of AdministrationSchedule objects describing the specified schedule.

Instructions

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

- Start with a basic request for this function, which includes logging in as the admin user and specifying the web service call to perform:

```
AdministrationServiceRequest rsr = new AdministrationServiceRequest();

rsr.setLoginId(this.username);
rsr.setPassword(this.password);
// This is the primary organisation
rsr.setOrgId(new Integer(1));

rsr.setFunction("LOADSCHEDULE");
```

- Use the parameters property to pass the schedule UUID:

```
// This is the Yellowfin Schedule UUID. Adjust this value
String[] parameters ={
"SOME_UUID"
};
rsr.setParameters(parameters);
```

- Once the request is configured, perform the call:

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

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

- The response will contain the following parameters: StatusCode and Schedules. (See details in the Response Parameters table above.)

Complete Example

Below is a full example of this web service call. To use it for yourself, carry out the following the steps:

1. Copy the code and save it as ws_loadschedule.jsp.
2. Put the file in the root folder: *Yellowfin/appserver/webapps/ROOT*.
3. Adjust the host, port, admin user, and schedule UUID values according to your environment.
4. Run *http://<host>:<port>/ws_loadschedule.jsp* from your Internet browser.

```
/*
 * LOADSCHEDULE Example.          ws_loadschedule.jsp.
 * A more complete example can be found in ws_admin_schedule_management.jsp
 */

<%@ page language="java" contentType="text/html; charset=UTF-8"
<%@ page import="java.text.*"
<%@ page import="java.util.*"
<%@ page import="com.hof.mi.web.service.*"
<%@ page import="com.hof.mi.web.service.schedule.*"
<%@ page import="com.hof.web.form.*"

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 web services admin account
rsr.setPassword("test");                           // change to the password of the above account
rsr.setOrgId(1);
rsr.setFunction("LOADSCHEDULE");

// existing Schedule UUID to load. Adjust this value
String[] parameters = {
        "SOME_UUID"
};
rsr.setParameters(parameters);


AdministrationServiceResponse rs = adminService.remoteAdministrationCall(rsr);

if ("SUCCESS".equals(rs.getStatusCode()) ) {
        AdministrationSchedule schedule = rs.getSchedule();
        out.write("Loaded schedule: " + schedule.getScheduleUUID() + "<br>");

        out.write("Schedule Type: " + schedule.getScheduleTypeCode() + "<br>");
        out.write("Description: " + schedule.getScheduleDescription() + "<br>");
        out.write("Is Active: " + schedule.isScheduleActive() + "<br>");
        out.write("Last Run Status: " + schedule.getLastRunStatus() + "<br>");
        out.write("Last Run Error: " + schedule.getLastRunError() + "<br>");
        out.write("Last Run Date: " + schedule.getLastRunDateTimeGMT() + "<br>");
        out.write("Next Run Date: " + schedule.getNextRunDateTimeGMT() + "<br>");

        // Some schedule types have extra information that you can access, see reference for details
        if (schedule instanceof ReportRefreshSchedule) {
                ReportRefreshSchedule rrs = (ReportRefreshSchedule)schedule;
                out.write("Report To Refresh: " + rrs.getReportId() + "<br>");
        }

        // these values all have different meanings depending on FrequencyType, see reference for details
        out.write("Frequency Type: " + sched.getFrequency().getFrequencyTypeCode() + "<br>");
        out.write("Frequency Code: " + sched.getFrequency().getFrequencyCode() + "<br>");
        out.write("Frequency Unit: " + sched.getFrequency().getFrequencyUnit() + "<br>");
        out.write("Frequency Local Time: " + sched.getFrequency().getLocalRunTime() + "<br>");
        out.write("Frequency Local Timezone: " + sched.getFrequency().getLocalTimezoneCode() + "<br>");

} else {
    out.write("Failure");
    out.write(" Code: " + rs.getErrorCode() );
}
```

This web service saves a single schedule. It uses the AdministrationSchedule object to pass the details of the schedule that need to be saved. The response will return this object with the new details. Note, that this does not create a new schedule, but updates the details of an existing one.

Request Parameters

The following parameters should be passed with this request:

Request Element	Data Type	Description
LoginId	String	An admin account to connect to Yellowfin web services. This can be the user ID or the email address, depending on the Logon ID method. This 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 service function. Set this to "SAVESCHEDULE".
Schedules	AdministrationSchedule	Object containing details of the schedule that is to be saved. See table below.

The following parameters must be included in the AdministrationSchedule object for this function:

AdministrationSchedule Element	Data Type	Description
ScheduleUUID	String	UUID of the schedule to be added.
Frequency	ScheduleFrequency	ScheduleFrequency object which defines the editable frequency options for this schedule. See table below.

For the Frequency object, exact required values are different for each ScheduleFrequency type, but all frequency types will contain the following fields:

Parameter	Data Type	Description
FrequencyTypeCode	String	Describes how to interpret the frequency information.
FrequencyCode	String	Has different meanings depending on the frequency type code.
FrequencyUnit	Integer	Has different meanings depending on the frequency type code
LocalRunTime	Integer	Seconds from midnight that this schedule will run on the given scheduled day.
LocalTimezoneCode	String	Java timezone offset code for the schedule to run.

Response Parameters

The returned response will contain these parameters:

Response Element	Data Type	Description
StatusCode	String	Status of the web service call. Possible values include: • SUCCESS • FAILURE
Schedules	AdministrationSchedule	This object will contain the updated details of the schedule.

Instructions

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

- Start with a basic request for this function, which includes logging in as the admin user and specifying the web service call to perform:

```
AdministrationServiceRequest rsr = new AdministrationServiceRequest();

rsr.setLoginId(this.username);
rsr.setPassword(this.password);
// This is the primary organisation
rsr.setOrgId(new Integer(1));

rsr.setFunction("SAVESCHEDULE");
```

- When updating a schedule, usually you would first load the schedule and then update it, but the only writable items are the Frequency and isActive parameters, so loading first is not required if you already know the Schedule UUID:

```
// This is the AdministrationSchedule which should be saved
AdministrationSchedule s = new AdministrationSchedule();
s.setScheduleUUID("SOME_KNOWN_EXISTING_UUID");
s.setActive(true);

// define the frequency information
ScheduleFrequency f = new MinutesFrequency();
f.setMinutes(5);
s.setFrequency(f);

// set the schedule in the request
rsr.setSchedule(s);
```

- Once the request is configured, perform the call:

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

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

- The response will contain the following parameters: StatusCode and Schedules. (See details in the Response Parameters table above.)

Complete Example

Below is a full example of this web service call. To use it for yourself, carry out the following the steps:

1. Copy the code and save it as ws_ saveschedule.jsp.
2. Put the file in the root folder: *Yellowfin/appserver/webapps/ROOT*.
3. Adjust the host, port, admin user values according to your environment.
4. Run *http://<host>:<port>/ws_ saveschedule.jsp* from your Internet browser.

```
/*
 * SAVESCHEDULE Example.    ws_ saveschedule.jsp.
 * A more complete example can be found in ws_admin_schedule_management.jsp
 */

<%@ page language="java" contentType="text/html; charset=UTF-8"
<%@ page import="java.text.*"
<%@ page import="java.util.*"
<%@ page import="com.hof.mi.web.service.*"
```

```
<%@ page import="com.hof.mi.web.service.schedule.*"
<%@ page import="com.hof.web.form.*"

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 web services admin account
rsr.setPassword("test");                           // change to the password of the above account
rsr.setOrgId(1);
rsr.setFunction("SAVESCHEDULE");

// normally you would load a schedule first and do some sort of modification
AdministrationSchedule editingSchedule = new AdministrationSchedule();
editingSchedule.setScheduleUUID("SOME_UUID");
editingSchedule.setScheduleActive(true);

ScheduleFrequency newFreq = new WeeklyFrequency();
newFreq.setDayOfWeek(ScheduleFrequency.MONDAY);
newFreq.setLocalRunTime(3 * 60 * 60); // 9am
newFreq.setLocalTimezoneCode("AUSTRALIA/SYDNEY");

editingSchedule.setFrequency(newFreq);
rsr.setSchedule(editingSchedule);

AdministrationServiceResponse rs = adminService.remoteAdministrationCall(rsr);

if ("SUCCESS".equals(rs.getStatusCode()) ) {
        AdministrationSchedule updatedSchedule = rs.getSchedule();
        out.write("Loaded schedule: " + updatedSchedule.getScheduleUUID() + "<br>");

        out.write("Schedule Type: " + updatedSchedule.getScheduleTypeCode() + "<br>");
        out.write("Description: " + updatedSchedule.getScheduleDescription() + "<br>");
        out.write("Is Active: " + updatedSchedule.isScheduleActive() + "<br>");
        out.write("Last Run Status: " + updatedSchedule.getLastRunStatus() + "<br>");
        out.write("Last Run Error: " + updatedSchedule.getLastRunError() + "<br>");
        out.write("Last Run Date: " + updatedSchedule.getLastRunDateTimeGMT() + "<br>");
        out.write("Next Run Date: " + updatedSchedule.getNextRunDateTimeGMT() + "<br>");

        // Some schedule types have extra information that you can access, see reference for details
        if (schedule instanceof ReportRefreshSchedule) {
                ReportRefreshSchedule rrs = (ReportRefreshSchedule)schedule;
                out.write("Report To Refresh: " + rrs.getReportId() + "<br>");
        }

        // these values all have different meanings depending on FrequencyType, see reference for details
        out.write("Frequency Type: " + updatedSchedule.getFrequency().getFrequencyTypeCode() + "<br>");
        out.write("Frequency Code: " + updatedSchedule.getFrequency().getFrequencyCode() + "<br>");
        out.write("Frequency Unit: " + updatedSchedule.getFrequency().getFrequencyUnit() + "<br>");
        out.write("Frequency Local Time: " + updatedSchedule.getFrequency().getLocalRunTime() + "<br>");
        out.write("Frequency Local Timezone: " + updatedSchedule.getFrequency().getLocalTimezoneCode() +
"<br>");

} else {
    out.write("Failure");
    out.write(" Code: " + rs.getErrorCode() );
}
```

This web service is used to delete a specified schedule.

Request Parameters

The following parameters should be passed with this request:

Request Element	Data Type	Description
LoginId	String	An admin account to connect to Yellowfin web services. This can be the user ID or the email address, depending on the Logon ID method. This 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 service function. Set this to "DELETESCHEDULE".
Parameters	String[]	The UUID of the schedule to be deleted. This should be set as the first element in the passed parameters array. This must be an existing UUID.

Request Example

Below is a SOAP XML example for this request:

```
<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>DELETESCHEDULE</function>
            <parameters>
                    <string>75a2f5b5-162b-49b5-b197-53643f7dc0de</string>
            </parameters>
         </arg0>
      </web:remoteAdministrationCall>
   </soapenv:Body>
</soapenv:Envelope>
```

Response Parameters

The returned response will contain these parameters:

Response Element	Data Type	Description
StatusCode	String	Status of the web service call. Possible values include: • SUCCESS • FAILURE

Response Example

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>
            <messages>Successfully Authenticated User: admin@yellowfin.com.au</messages>
            <messages>Web Service Request Complete</messages>
            <sessionId>7b8e70f20d25079f86cf26f5712d15f9</sessionId>
            <statusCode>SUCCESS</statusCode>
         </return>
      </ns2:remoteAdministrationCallResponse>
   </S:Body>
</S:Envelope>
```

Instructions

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

- Start with a basic request for this function, which includes logging in as the admin user and specifying the web service call to perform:

  ```
  AdministrationServiceRequest rsr = new AdministrationServiceRequest();

  rsr.setLoginId(this.username);
  rsr.setPassword(this.password);
  // This is the primary organisation
  rsr.setOrgId(new Integer(1));

  rsr.setFunction("DELETESCHEDULE");
  ```

- Specify an existing schedule to delete it:

  ```
  // This is the Yellowfin Schedule UUID
  String[] parameters ={
  "SOME_UUID"
  };
  rsr.setParameters(parameters);
  ```

- Once the request is configured, perform the call:

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

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

- The response will contain the StatusCode parameter. (See details in the Response Parameters table above.)

Complete Example

Below is a full example of this web service call. To use it for yourself, carry out the following the steps:

1. Copy the code and save it as ws_ deleteschedule.jsp.
2. Put the file in the root folder: *Yellowfin/appserver/webapps/ROOT*.
3. Adjust the host, port, admin user, and schedule UUID values according to your environment.
4. Run *http://<host>:<port>/ws_ deleteschedule.jsp* from your Internet browser.

```
/*
 * DELETESCHEDULE Example.        ws_deleteschedule.jsp
 * A more complete example can be found in ws_admin_schedule_management.jsp
 */

<%@ page language="java" contentType="text/html; charset=UTF-8"
<%@ page import="java.text.*"
<%@ page import="java.util.*"
<%@ page import="com.hof.mi.web.service.*"
<%@ page import="com.hof.mi.web.service.schedule.*"
<%@ page import="com.hof.web.form.*"

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 web services admin account
rsr.setPassword("test");                           // change to the password of the above account
rsr.setOrgId(1);
rsr.setFunction("DELETESCHEDULE");

String scheduleUUID = "SOME_UUID";
// existing Schedule UUID to delete
String[] parameters = {
        scheduleUUID
};
rsr.setParameters(parameters);


AdministrationServiceResponse rs = adminService.remoteAdministrationCall(rsr);

if ("SUCCESS".equals(rs.getStatusCode()) ) {
        out.write("Successfully deleted schedule: " + scheduleUUID);
} else {
    out.write("Failure");
    out.write(" Code: " + rs.getErrorCode() );
}
```

This web service submits a schedule to run it.

Request Parameters

The following parameters should be passed with this request:

Request Element	Data Type	Description
LoginId	String	An admin account to connect to Yellowfin web services. This can be the user ID or the email address, depending on the Logon ID method. This 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 service function. Set this to "RUNSCHEDULENOW".

Parameters	String[]	The UUID of the schedule to be executed. This should be set as the first element in the passed parameters array. This must be an existing UUID.

Response Parameters

The returned response will contain these parameters:

Response Element	Data Type	Description
StatusCode	String	Status of the web service call. Possible values include: • SUCCESS • FAILURE
Schedules	AdministrationSchedule	This object will contain the details of the schedule requested to be run.

Instructions

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

- Start with a basic request for this function, which includes logging in as the admin user and specifying the web service call to perform:

```
AdministrationServiceRequest rsr = new AdministrationServiceRequest();

rsr.setLoginId(this.username);
rsr.setPassword(this.password);
// This is the primary organisation
rsr.setOrgId(new Integer(1));

rsr.setFunction("RUNSCHEDULENOW");
```

- Specify an existing schedule to run it:

```
// This is the Yellowfin Schedule UUID
String[] parameters ={
"SOME_UUID"
};
rsr.setParameters(parameters);
```

- Once the request is configured, perform the call:

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

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

- The response will contain the following parameters: StatusCode and Schedules. (See details in the Response Parameters table above.)

Complete Example

Below is a full example of this web service call. To use it for yourself, carry out the following the steps:

1. Copy the code and save it as ws_runschedulenow.jsp.
2. Put the file in the root folder: *Yellowfin/appserver/webapps/ROOT*.
3. Adjust the host, port, admin user, and schedule UUID values according to your environment.

4. Run *http://<host>:<port>/ws_runschedulenow.jsp* from your Internet browser.

```
/*
 * RUNSCHEDULENOW Example.     ws_runschedulenow.jsp
 * A more complete example can be found in ws_admin_schedule_management.jsp
 */

<%@ page language="java" contentType="text/html; charset=UTF-8"
<%@ page import="java.text.*"
<%@ page import="java.util.*"
<%@ page import="com.hof.mi.web.service.*"
<%@ page import="com.hof.mi.web.service.schedule.*"
<%@ page import="com.hof.web.form.*"

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 web services admin account
rsr.setPassword("test");                           // change to the password of the above account
rsr.setOrgId(1);
rsr.setFunction("RUNSCHEDULENOW");

// existing Schedule UUID to submit for running
String scheduleUUID = "SOME_UUID";
String[] parameters = {
        scheduleUUID
};
rsr.setParameters(parameters);


AdministrationServiceResponse rs = adminService.remoteAdministrationCall(rsr);

if ("SUCCESS".equals(rs.getStatusCode()) ) {
        out.write("Successfully submitted schedule " + scheduleUUID + " to be run.");

        // NOTE: some properties like last run status/date, etc will not yet be updated,
        // since this call only submits the schedule for run and does not wait until it is complete.
        AdministrationSchedule updatedSchedule = rs.getSchedule();
        out.write("Loaded schedule: " + updatedSchedule.getScheduleUUID() + "<br>");

        out.write("Schedule Type: " + updatedSchedule.getScheduleTypeCode() + "<br>");
        out.write("Description: " + updatedSchedule.getScheduleDescription() + "<br>");
        out.write("Is Active: " + updatedSchedule.isScheduleActive() + "<br>");
        out.write("Last Run Status: " + updatedSchedule.getLastRunStatus() + "<br>");
        out.write("Last Run Error: " + updatedSchedule.getLastRunError() + "<br>");
        out.write("Last Run Date: " + updatedSchedule.getLastRunDateTimeGMT() + "<br>");
        out.write("Next Run Date: " + updatedSchedule.getNextRunDateTimeGMT() + "<br>");

        // Some schedule types have extra information that you can access, see reference for details
        if (schedule instanceof ReportRefreshSchedule) {
                ReportRefreshSchedule rrs = (ReportRefreshSchedule)schedule;
                out.write("Report To Refresh: " + rrs.getReportId() + "<br>");
        }

        // these values all have different meanings depending on FrequencyType, see reference for details
        out.write("Frequency Type: " + updatedSchedule.getFrequency().getFrequencyTypeCode() + "<br>");
        out.write("Frequency Code: " + updatedSchedule.getFrequency().getFrequencyCode() + "<br>");
        out.write("Frequency Unit: " + updatedSchedule.getFrequency().getFrequencyUnit() + "<br>");
        out.write("Frequency Local Time: " + updatedSchedule.getFrequency().getLocalRunTime() + "<br>");
        out.write("Frequency Local Timezone: " + updatedSchedule.getFrequency().getLocalTimezoneCode() +
"<br>");


} else {
    out.write("Failure");
    out.write(" Code: " + rs.getErrorCode() );
}
```

This web service pauses a schedule. Here “pausing” a schedule refers to pausing it’s frequency schedule and does not affect the schedule if it is currently running or queued for running, but will stop it from being queued in the future.

NOTE: This is a shorthand way of pausing the schedule. This can also be achieved by loading the schedule, updating it’s active state to false and then saving it.

Request Parameters

The following parameters should be passed with this request:

Request Element	Data Type	Description
LoginId	String	An admin account to connect to Yellowfin web services. This can be the user ID or the email address, depending on the Logon ID method. This 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 service function. Set this to "PAUSESCHEDULE".
Parameters	String[]	The UUID of the schedule to be paused. This should be set as the first element in the passed parameters array. This must be an existing UUID.

Response Parameters

The returned response will contain these parameters:

Response Element	Data Type	Description
StatusCode	String	Status of the web service call. Possible values include: • SUCCESS • FAILURE
Schedules	AdministrationSchedule	This object will contain the details of the schedule requested to be paused.

Instructions

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

- Start with a basic request for this function, which includes logging in as the admin user and specifying the web service call to perform:

```
AdministrationServiceRequest rsr = new AdministrationServiceRequest();

rsr.setLoginId(this.username);
rsr.setPassword(this.password);
// This is the primary organisation
rsr.setOrgId(new Integer(1));

rsr.setFunction("PAUSESCHEDULE");
```

- Specify an existing schedule to pause it's schedule frequency:

```
// This is the Yellowfin Schedule UUID
String[] parameters ={
"SOME_UUID"
};
rsr.setParameters(parameters);
```

- Once the request is configured, perform the call:

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

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

- The response will contain the following parameters: StatusCode and Schedules. (See details in the Response Parameters table above.)

Complete Example

Below is a full example of this web service call. To use it for yourself, carry out the following the steps:

1. Copy the code and save it as ws_pauseschedule.jsp.
2. Put the file in the root folder: *Yellowfin/appserver/webapps/ROOT*.
3. Adjust the host, port, admin user, and schedule UUID values according to your environment.
4. Run *http://<host>:<port>/ws_pauseschedule.jsp* from your Internet browser.

```
/*
 * PAUSESCHEDULE Example.                ws_pauseschedule.jsp
 * A more complete example can be found in ws_admin_schedule_management.jsp
 */

<%@ page language="java" contentType="text/html; charset=UTF-8"
<%@ page import="java.text.*"
<%@ page import="java.util.*"
<%@ page import="com.hof.mi.web.service.*"
<%@ page import="com.hof.mi.web.service.schedule.*"
<%@ page import="com.hof.web.form.*"

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 web services admin account
rsr.setPassword("test");                           // change to the password of the above account
rsr.setOrgId(1);
rsr.setFunction("PAUSESCHEDULE");

// existing Schedule UUID to submit for running
String scheduleUUID = "SOME_UUID";
String[] parameters = {
        scheduleUUID
};
rsr.setParameters(parameters);


AdministrationServiceResponse rs = adminService.remoteAdministrationCall(rsr);

if ("SUCCESS".equals(rs.getStatusCode()) ) {
        out.write("Successfully paused schedule " + scheduleUUID);

        AdministrationSchedule updatedSchedule = rs.getSchedule();
        out.write("Loaded schedule: " + updatedSchedule.getScheduleUUID() + "<br>");

        out.write("Schedule Type: " + updatedSchedule.getScheduleTypeCode() + "<br>");
        out.write("Description: " + updatedSchedule.getScheduleDescription() + "<br>");
        out.write("Is Active: " + updatedSchedule.isScheduleActive() + "<br>");
        out.write("Last Run Status: " + updatedSchedule.getLastRunStatus() + "<br>");
        out.write("Last Run Error: " + updatedSchedule.getLastRunError() + "<br>");
        out.write("Last Run Date: " + updatedSchedule.getLastRunDateTimeGMT() + "<br>");
        out.write("Next Run Date: " + updatedSchedule.getNextRunDateTimeGMT() + "<br>");

        // Some schedule types have extra information that you can access, see reference for details
        if (schedule instanceof ReportRefreshSchedule) {
                ReportRefreshSchedule rrs = (ReportRefreshSchedule)schedule;
                out.write("Report To Refresh: " + rrs.getReportId() + "<br>");
        }

        // these values all have different meanings depending on FrequencyType, see reference for details
        out.write("Frequency Type: " + updatedSchedule.getFrequency().getFrequencyTypeCode() + "<br>");
        out.write("Frequency Code: " + updatedSchedule.getFrequency().getFrequencyCode() + "<br>");
        out.write("Frequency Unit: " + updatedSchedule.getFrequency().getFrequencyUnit() + "<br>");
        out.write("Frequency Local Time: " + updatedSchedule.getFrequency().getLocalRunTime() + "<br>");
        out.write("Frequency Local Timezone: " + updatedSchedule.getFrequency().getLocalTimezoneCode() +
"<br>");


} else {
    out.write("Failure");
    out.write(" Code: " + rs.getErrorCode() );
}
```

This web service resumes a paused schedule. “Resuming” a schedule refers to resuming it’s frequency schedule and does not queue the item for running, but will allow it to be queued for running again upon its next scheduled date/time.

NOTE: This is a shorthand way of resuming the schedule. This can also be achieved by loading the schedule, updating it’s active state to true and then saving it.

Request Parameters

The following parameters should be passed with this request:

Request Element	Data Type	Description
LoginId	String	An admin account to connect to Yellowfin web services. This can be the user ID or the email address, depending on the Logon ID method. This 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 service function. Set this to "RESUMESCHEDULE".
Parameters	String[]	The UUID of the schedule to be resumed. This should be set as the first element in the passed parameters array. This must be an existing UUID.

Response Parameters

The returned response will contain these parameters:

Response Element	Data Type	Description
StatusCode	String	Status of the web service call. Possible values include: • SUCCESS • FAILURE
Schedules	AdministrationSchedule	This object will contain the details of the schedule requested to be resumed.

Instructions

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

- Start with a basic request for this function, which includes logging in as the admin user and specifying the web service call to perform:

```
AdministrationServiceRequest rsr = new AdministrationServiceRequest();

rsr.setLoginId(this.username);
rsr.setPassword(this.password);
// This is the primary organisation
rsr.setOrgId(new Integer(1));

rsr.setFunction("RESUMESCHEDULE");
```

- Specify an existing schedule to resume its schedule frequency it:

```
// This is the Yellowfin Schedule UUID
String[] parameters ={
"SOME_UUID"
};
rsr.setParameters(parameters);
```

- Once the request is configured, perform the call:

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

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

- The response will contain the following parameters: StatusCode and Schedules. (See details in the Response Parameters table above.)

Complete Example

Below is a full example of this web service call. To use it for yourself, carry out the following the steps:

1. Copy the code and save it as ws_resumeschedule.jsp.
2. Put the file in the root folder: *Yellowfin/appserver/webapps/ROOT*.
3. Adjust the host, port, admin user, and schedule UUID values according to your environment.
4. Run *http://<host>:<port>/ws_resumeschedule.jsp* from your Internet browser.

```
/*
 * RESUMESCHEDULE Example.                        ws_resumeschedule.jsp
 * A more complete example can be found in ws_admin_schedule_management.jsp
 */

<%@ page language="java" contentType="text/html; charset=UTF-8"
<%@ page import="java.text.*"
<%@ page import="java.util.*"
<%@ page import="com.hof.mi.web.service.*"
<%@ page import="com.hof.mi.web.service.schedule.*"
<%@ page import="com.hof.web.form.*"

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 web services admin account
rsr.setPassword("test");                           // change to the password of the above account
rsr.setOrgId(1);
rsr.setFunction("RESUMESCHEDULE");

// existing Schedule UUID to submit for running
String scheduleUUID = "SOME_UUID";
String[] parameters = {
        scheduleUUID
};
rsr.setParameters(parameters);


AdministrationServiceResponse rs = adminService.remoteAdministrationCall(rsr);

if ("SUCCESS".equals(rs.getStatusCode()) ) {
        out.write("Successfully resumed schedule " + scheduleUUID);

        AdministrationSchedule updatedSchedule = rs.getSchedule();
        out.write("Loaded schedule: " + updatedSchedule.getScheduleUUID() + "<br>");

        out.write("Schedule Type: " + updatedSchedule.getScheduleTypeCode() + "<br>");
        out.write("Description: " + updatedSchedule.getScheduleDescription() + "<br>");
        out.write("Is Active: " + updatedSchedule.isScheduleActive() + "<br>");
        out.write("Last Run Status: " + updatedSchedule.getLastRunStatus() + "<br>");
        out.write("Last Run Error: " + updatedSchedule.getLastRunError() + "<br>");
        out.write("Last Run Date: " + updatedSchedule.getLastRunDateTimeGMT() + "<br>");
        out.write("Next Run Date: " + updatedSchedule.getNextRunDateTimeGMT() + "<br>");

        // Some schedule types have extra information that you can access, see reference for details
        if (schedule instanceof ReportRefreshSchedule) {
                ReportRefreshSchedule rrs = (ReportRefreshSchedule)schedule;
                out.write("Report To Refresh: " + rrs.getReportId() + "<br>");
        }

        // these values all have different meanings depending on FrequencyType, see reference for details
        out.write("Frequency Type: " + updatedSchedule.getFrequency().getFrequencyTypeCode() + "<br>");
        out.write("Frequency Code: " + updatedSchedule.getFrequency().getFrequencyCode() + "<br>");
        out.write("Frequency Unit: " + updatedSchedule.getFrequency().getFrequencyUnit() + "<br>");
        out.write("Frequency Local Time: " + updatedSchedule.getFrequency().getLocalRunTime() + "<br>");
        out.write("Frequency Local Timezone: " + updatedSchedule.getFrequency().getLocalTimezoneCode() +
"<br>");


} else {
    out.write("Failure");
    out.write(" Code: " + rs.getErrorCode() );
}
```