# Content Export and Import Services

The web service calls categorized in this section are related to exporting and importing content. The web service API is currently limited to exporting or importing the following types of content:

- Report
- Report category
- Report subcategory
- Data source
- View
- Dashboard
- Transformation flow

## Export Web Services

This process will help you navigate through the export web services, to get exportable information and reuse it for other services:

1. Use the GETCONTENT function to retrieve all the Yellowfin content available to be exported. This will also be useful if you want to export everything.
2. This function returns an array of ContentResource objects containing Yellowfin content details, that you can reuse in other calls to import, export, or validate details. Refer to the object definitions for more details on this object.
3. You can also configure your own export list and place content definitions into the ContentResource object. The object definition will reveal what elements are required for each Yellowfin content.
4. You need to know the object's **resourceType** value for each of the Yellowfin content types:

| Content type | ContentResource resourceType |
|---|---|
| Report category | RPTCATEGORY |
| Report subcategory | RPTSUBCATEGORY |
| Data source | DATASOURCE |
| View | VIEW |
| Dashboard | GROUP |
| Report | REPORT |
| Data Transformation | ETLPROCESS |

5. Instead of manually searching for dependencies of all content types, use the GETEXPORTDEPENDENCIES function, specifying a content whose dependencies you need by storing its details in the ContentResource object.
6. For instance, if you export one dashboard, you can include a single object into **ContentResource**, representing that particular dashboard. The function will return that dashboard's dependencies, including its reports, views, data sources, categories, and sub categories (these are returned in a ContentResource object array).
7. To get an XML file with Yellowfin content, you can create an array of multiple ContentResource objects and call the EXPORTCONTENT function. You can proceed to import this file into another Yellowfin environment as well.
8. Note:  The GETEXPORTDEPENDENCIES and EXPORTCONTENT web services do not work properly with Client Org resources. Only default org resources can be exported properly using these functions.

### Main Export Functions

This function returns all Yellowfin content that can be exported.

#### 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.<br><br>This account must have the “web services” role enabled, and must belong to the default (ie, primary) org. |
| Password | String | Password of the above account. |
| OrgId | Integer | Default (ie, primary) organization ID within Yellowfin. Always set this to 1. |
| Function | String | Web service function. Set this to "GETCONTENT". |
| OrgRef | String | This optional parameter can be used to specify a client org. ID |

### 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>GETCONTENT</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:<br><br>• SUCCESS<br>• FAILURE |
| ContentResources | ContentResource[] | Object array containing details of Yellowfin's content that can be exported. |

### 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>
            <contentResources>
               <resourceCode>AUDITREPORTS</resourceCode>
               <resourceDescription>Audit Reports</resourceDescription>
               <resourceId>56339</resourceId>
               <resourceName>Audit Reports</resourceName>
               <resourceOrgId>1</resourceOrgId>
               <resourceType>RPTCATEGORY</resourceType>
               <resourceUUID>a6bdc6b5-a832-42a2-98c7-18273900d0aa</resourceUUID>
            </contentResources>
```

```
<contentResources>
   <resourceCode>ADMINREPORTS</resourceCode>
   <resourceDescription>Admin Reports</resourceDescription>
   <resourceId>56340</resourceId>
   <resourceName>Admin Reports</resourceName>
   <resourceOrgId>1</resourceOrgId>
   <resourceType>RPTSUBCATEGORY</resourceType>
   <resourceUUID>f7fb32b7-1573-4899-916f-c34afb9a865d</resourceUUID>
</contentResources>
<contentResources>
   <resourceCode>CONTENTUSAGE</resourceCode>
   <resourceDescription>Content Usage</resourceDescription>
   <resourceId>56341</resourceId>
   <resourceName>Content Usage</resourceName>
   <resourceOrgId>1</resourceOrgId>
   <resourceType>RPTSUBCATEGORY</resourceType>
   <resourceUUID>6bae5230-c1f9-4491-8a8b-f14b1ae660d7</resourceUUID>
</contentResources>
<contentResources>
   <resourceCode>USERACCESS</resourceCode>
   <resourceDescription>User Access</resourceDescription>
   <resourceId>56342</resourceId>
   <resourceName>User Access</resourceName>
   <resourceOrgId>1</resourceOrgId>
   <resourceType>RPTSUBCATEGORY</resourceType>
   <resourceUUID>0c7ddde4-fa03-4e88-b37b-7b5e4aad5e1d</resourceUUID>
</contentResources>
<contentResources>
   <resourceCode>TUTORIAL</resourceCode>
   <resourceDescription>Tutorial</resourceDescription>
   <resourceId>60706</resourceId>
   <resourceName>Tutorial</resourceName>
   <resourceOrgId>1</resourceOrgId>
   <resourceType>RPTCATEGORY</resourceType>
   <resourceUUID>a23c2ec6-a2fa-45c7-b5da-dcf3f02e6633</resourceUUID>
</contentResources>
<contentResources>
   <resourceCode>ATHLETES</resourceCode>
   <resourceDescription>Athletes</resourceDescription>
   <resourceId>60707</resourceId>
   <resourceName>Athletes</resourceName>
   <resourceOrgId>1</resourceOrgId>
   <resourceType>RPTSUBCATEGORY</resourceType>
   <resourceUUID>72e4b4bd-a482-4a01-a031-c6ab76dbb3a5</resourceUUID>
</contentResources>
<contentResources>
   <resourceCode>CAMP</resourceCode>
   <resourceDescription>Camp</resourceDescription>
   <resourceId>60708</resourceId>
   <resourceName>Camp</resourceName>
   <resourceOrgId>1</resourceOrgId>
   <resourceType>RPTSUBCATEGORY</resourceType>
   <resourceUUID>465411e5-594b-478e-af64-c0f59fc4546f</resourceUUID>
</contentResources>
<contentResources>
   <resourceCode>KPIS</resourceCode>
   <resourceDescription>KPIs</resourceDescription>
   <resourceId>60709</resourceId>
   <resourceName>KPIs</resourceName>
   <resourceOrgId>1</resourceOrgId>
   <resourceType>RPTSUBCATEGORY</resourceType>
   <resourceUUID>d514c643-dc01-4781-8905-d34e761ccd19</resourceUUID>
</contentResources>
<contentResources>
   <resourceCode>MARKETINGBOOKING</resourceCode>
   <resourceDescription>Marketing &amp; Booking</resourceDescription>
   <resourceId>60710</resourceId>
   <resourceName>Marketing &amp; Booking</resourceName>
   <resourceOrgId>1</resourceOrgId>
   <resourceType>RPTSUBCATEGORY</resourceType>
   <resourceUUID>dbe6d0a3-c088-4d71-b65a-f383aaa54be9</resourceUUID>
```

```
            </contentResources>
            <contentResources>
               <resourceCode>TRAINING</resourceCode>
               <resourceDescription>Training</resourceDescription>
               <resourceId>60711</resourceId>
               <resourceName>Training</resourceName>
               <resourceOrgId>1</resourceOrgId>
               <resourceType>RPTSUBCATEGORY</resourceType>
               <resourceUUID>c503ea57-cc69-43a9-98bc-a90ebbe1c864</resourceUUID>
            </contentResources>
            <contentResources>
               <resourceDescription/>
               <resourceId>70101</resourceId>
               <resourceName>Oracle database</resourceName>
               <resourceOrgId>1</resourceOrgId>
               <resourceType>DATASOURCE</resourceType>
            </contentResources>
            <contentResources>
               <resourceDescription/>
               <resourceId>70109</resourceId>
               <resourceName>Oracle</resourceName>
               <resourceOrgId>1</resourceOrgId>
               <resourceType>DATASOURCE</resourceType>
            </contentResources>
            <contentResources>
               <resourceDescription>Ski Team is the demonstration and tutorial database.</resourceDescription>
               <resourceId>54700</resourceId>
               <resourceName>Ski Team</resourceName>
               <resourceOrgId>1</resourceOrgId>
               <resourceType>DATASOURCE</resourceType>
            </contentResources>
            <contentResources>
               <resourceDescription/>
               <resourceId>54701</resourceId>
               <resourceName>Yellowfin Configuration Database</resourceName>
               <resourceOrgId>1</resourceOrgId>
               <resourceType>DATASOURCE</resourceType>
            </contentResources>
            <contentResources>
               <resourceDescription/>
               <resourceId>70108</resourceId>
               <resourceName>c</resourceName>
               <resourceOrgId>1</resourceOrgId>
               <resourceType>DATASOURCE</resourceType>
            </contentResources>
            <contentResources>
               <resourceDescription/>
               <resourceId>70110</resourceId>
               <resourceName>f</resourceName>
               <resourceOrgId>1</resourceOrgId>
               <resourceType>DATASOURCE</resourceType>
            </contentResources>
            <contentResources>
               <resourceDescription>This is a tutorial view for training and demo purposes.<
/resourceDescription>
               <resourceId>60543</resourceId>
               <resourceName>Ski Team</resourceName>
               <resourceType>VIEW</resourceType>
               <resourceUUID>e3632adb-5194-460c-a172-c085416f493f</resourceUUID>
            </contentResources>
            <contentResources>
               <resourceDescription>This view should be used to monitor usage of Yellowfin by User and Content
Type.</resourceDescription>
               <resourceId>56169</resourceId>
               <resourceName>Yellowfin Usage Audit</resourceName>
               <resourceType>VIEW</resourceType>
               <resourceUUID>fb6416c4-441e-42b3-a442-e7426f25f6b4</resourceUUID>
            </contentResources>
            <contentResources>
               <resourceDescription>This dashboard contains a set of reports covering general system and admin
information, including performance, sessions, data source &amp; view usage.</resourceDescription>
```

```
                <resourceId>57438</resourceId>
                <resourceName>Admin</resourceName>
                <resourceOrgId>1</resourceOrgId>
                <resourceType>GROUP</resourceType>
                <resourceUUID>33827292-cda6-4071-965f-730ccbc53519</resourceUUID>
            </contentResources>
            <contentResources>
                <resourceDescription>This is an analytic tab that is used to understand examine metrics split by
various demographics and filters.</resourceDescription>
                <resourceId>61195</resourceId>
                <resourceName>Analysis</resourceName>
                <resourceOrgId>1</resourceOrgId>
                <resourceType>GROUP</resourceType>
                <resourceUUID>f19e63f5-7175-4c57-897d-ed865aba8972</resourceUUID>
            </contentResources>
            </contentResources>
            <errorCode>0</errorCode>
            <messages>Successfully Authenticated User: admin@yellowfin.com.au</messages>
            <messages>Web Service Request Complete</messages>
            <sessionId>92029c8ae4f1db0f33bf0b7370c1088a</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:

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

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

- You may even identify a specific client organization:

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

- 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 elements: StatusCode and ContentResource. (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_ getcontent.jsp.

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

```
<%
/*            ws_getcontent.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.*" %>
<%

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("GETCONTENT");

AdministrationServiceResponse rs = adminService.remoteAdministrationCall(rsr);
                if ("SUCCESS".equals(rs.getStatusCode()) ) {
                            out.write("<br>Success");
                            ContentResource[] crs = rs.getContentResources();
                            out.write("<table>");
                            out.write("<tr><td> id </td><td> type </td><td> UUID </td></tr>");
                            for (ContentResource c: crs) {
                                        out.write("<tr>");
                                        out.write("<td>" + c.getResourceId() + "</td><td>" + c.getResourceType()
+ "</td><td>" + c.getResourceUUID() + "</td>");
                                        out.write("</tr>");
                            }
                }
                else {
                out.write("<br>Failure");
                out.write(" Code: " + rs.getErrorCode());
                }
%>
```

---

This function exports selected Yellowfin content into an XML file.

### 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.<br><br>This account must have the “web services” role enabled, and must belong to the default (ie, primary) org. |
| Password | String | Password of the above account. |

| | | |
|---|---|---|
| OrgId | Integer | Default (ie, primary) organization ID within Yellowfin. Always set this to 1. |
| Function | String | Web service function. Set this to "EXPORTCONTENT". |
| OrgRef | String | This optional parameter can be used to specify a client org. ID. |
| ContentResources | ContentResource | Object used to specify the content that is to be exported. See table below. |

The following parameters are specified in the **ContentResource** object to call this function:

| ContentResource Element | Data Type | Description |
|---|---|---|
| ResourceID | Integer | Mandatory parameter to provide internal ID of the content. |
| ResourceType | String | Mandatory parameter to specify the content type. Could be one of:<br>• RPTCATEGORY<br>• RPTSUBCATEGORY<br>• DATASOURCE<br>• VIEW<br>• GROUP<br>• REPORT<br>• ETLPROCESS |
| ResourceUUID | String | This optional parameter can be used to provide the UUID of the content. |

### 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>EXPORTCONTENT</function>
            <contentResources>
               <resourceId>56169</resourceId>
               <resourceType>VIEW</resourceType>
            </contentResources>
         </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:<br>• SUCCESS<br>• FAILURE |
| BinaryAttachments | ReportBinaryObject[] | Object array containing details of Yellowfin's content that can be exported. See table below. |

The **ReportBinaryObject** array will return the following parameters with this call:

| ReportBinaryObject Element | Data Type | Description |
|---|---|---|
| Key | String | The unique key for binary object storage for this function will be "EXPORT/XML". |
| ContentType | String | The MIME type for this function will be "text/XML". |
| Data | Byte[] | This array will contain the metadata of the content that can be saved into an XML file. |

### 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>
            <binaryAttachments>
               <contentType>text/xml</contentType>
               <data>PD94bFORERST1A8L.....</data>
               <key>EXPORTXML</key>
            </binaryAttachments>
            <errorCode>0</errorCode>
            <messages>Successfully Authenticated User: admin@yellowfin.com.au</messages>
            <messages>Web Service Request Complete</messages>
            <sessionId>111efc95cc598355645e6bf8d588d80f</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:

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

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

- Specify which content to export by using an object:

```
ContentResource[] cr = new ContentResource[1];
cr[0] = new ContentResource();
cr[0].setResourceId(70058);
cr[0].setResourceType("GROUP");
cr[0].setResourceOrgId(1);
```

- Place the object in your request:

```
rsr.setContentResources(cr);
```

- 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 elements: StatusCode and ReportBinaryObject. (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_exportcontent.jsp.
2. Put the file in the root folder: *Yellowfin/appserver/webapps/ROOT*.
3. Adjust the host, port, and admin user details according to your environment.
4. Run *http://<host>:<port>/ws_exportcontent.jsp* from your Internet browser.

```
<%
/*            ws_exportcontent.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.*" %>
<%@ page import="java.nio.file.Files" %>
<%@ page import="java.io.PrintWriter" %>
<%
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");                            // set to the password of the above account
rsr.setOrgId(1);
rsr.setFunction("EXPORTCONTENT");

// specify which dashboard to export:
ContentResource[] cr = new ContentResource[1];
cr[0] = new ContentResource();
cr[0].setResourceId(70058);
cr[0].setResourceType("GROUP");
cr[0].setResourceOrgId(1);

rsr.setContentResources(cr);

AdministrationServiceResponse rs = adminService.remoteAdministrationCall(rsr);

if ("SUCCESS".equals(rs.getStatusCode()) ) {
                out.write("<br>Success");
                byte[] data = rs.getBinaryAttachments()[0].getData();
                String xml = new String(data, "UTF-8");
                PrintWriter writer = new PrintWriter("/Applications/Yellowfin 7.4/YFexport.xml", "UTF-8");
                writer.println(xml);
                writer.close();

                ReportBinaryObject[] bo = rs.getBinaryAttachments();
                for (ReportBinaryObject o : bo){
                            out.write("<br><br>Key: " + o.getKey());
                            out.write("<br>Content Type: " + o.getContentType());
                }


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

---

This function returns all the dependencies of a specific content. The ContentResource object is used to specify the content with the help of the resource ID (which can be retrieved using the GETCONTENT call). For instance if a report is the defined content type, then the response will display its dependencies, such as the report category, sub category, data source, view, etc.

### 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.<br><br>This account must have the “web services” role enabled, and must belong to the default (ie, primary) org. |
| Password | String | Password of the above account. |
| OrgId | Integer | Default (ie, primary) organization ID within Yellowfin. Always set this to 1. |
| Function | String | Web service function. Set this to "GETEXPORTDEPENDENCIES". |
| OrgRef | String | This optional parameter can be used to specify a client org. ID. |
| ContentResources | ContentResource | Object containing metadata of the content whose dependencies are to be retrieved. See table below. |

The following parameters are specified in the ContentResource object to call this function:

| ContentResource Element | Data Type | Description |
|---|---|---|
| ResourceID | Integer | Mandatory parameter to provide internal ID of the content. |
| ResourceType | String | Mandatory parameter to specify the content type. Could be one of:<br><br>• RPTCATEGORY<br>• RPTSUBCATEGORY<br>• DATASOURCE<br>• VIEW<br>• GROUP<br>• REPORT<br>• ETLPROCESS |
| ResourceUUID | String | This optional parameter can be used to provide the UUID of the content. |

### 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>GETEXPORTDEPENDENCIES</function>
            <contentResources>
               <resourceId>56169</resourceId>
               <resourceType>VIEW</resourceType>
            </contentResources>
         </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:<br>• SUCCESS<br>• FAILURE |
| ContentResources | ContentResource[] | Object array containing metadata of the specified artifact's dependencies. |

### 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>
            <contentResources>
               <resourceDescription/>
               <resourceId>54701</resourceId>
               <resourceName>Yellowfin Configuration Database</resourceName>
               <resourceOrgId>1</resourceOrgId>
               <resourceType>DATASOURCE</resourceType>
            </contentResources>
            <errorCode>0</errorCode>
            <messages>Successfully Authenticated User: admin@yellowfin.com.au</messages>
            <messages>Web Service Request Complete</messages>
            <sessionId>97d7f893d787daf2806a13cdfa6f09d3</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:

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

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

- You may even identify a specific client organization:

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

- Using a ContentResource object, specify details of the content whose dependencies are to be retrieved:

```
ContentResource[] cr = new ContentResource[1];

cr[0] = new ContentResource();
cr[0].setResourceId(70307);
cr[0].setResourceType("GROUP");
cr[0].setResourceOrgId(1);
```

- Then set this object in the request:

```
rsr.setContentResources(cr);
```

- 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 elements: StatusCode and ContentResource. (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_ getexportdependencies.jsp.
2. Put the file in the root folder: *Yellowfin/appserver/webapps/ROOT*.
3. Adjust the host, port, and admin user details according to your environment.
4. Run *http://<host>:<port>/ws_ getexportdependencies.jsp* from your Internet browser.

```
<%
/*            ws_getexportdependecies.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.*" %>
<%@ page import="java.nio.file.Files" %>
<%@ page import="java.io.PrintWriter" %>
<%
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");                                // set to the password of the account above
rsr.setOrgId(1);
rsr.setFunction("GETEXPORTDEPENDENCIES");

ContentResource[] cr = new ContentResource[1];

cr[0] = new ContentResource();
cr[0].setResourceId(70307);
cr[0].setResourceType("GROUP");
cr[0].setResourceOrgId(1);

rsr.setContentResources(cr);

AdministrationServiceResponse rs = adminService.remoteAdministrationCall(rsr);

if ("SUCCESS".equals(rs.getStatusCode()) ) {
                out.write("<br>Success");
                ContentResource[] crs = rs.getContentResources();
                out.write("<table>");
                out.write("<tr><td> id </td><td> type </td><td> UUID </td></tr>");
                for (ContentResource c: crs) {
                            out.write("<tr>");
                            out.write("<td>" + c.getResourceId() + "</td><td>" + c.getResourceType() + "<
/td><td>" + c.getResourceUUID() + "</td>");
                            out.write("</tr>");
   }
} else {
                out.write("Failure");
                out.write(" Code: " + rs.getErrorCode());
}
%>
```

---

---

## Import Web Services

The following tips will enable you to navigate through Yellowfin's import web services:

1. In order to import content using a web service, you would need a YFX or XML file. This is retrieved by calling the EXPORTCONTENT function.
    - You can also use Yellowfin's export feature to do this. However, a file generated via this feature will contain more content types than those currently supported by the web service API.

- The following content types cannot be imported via web services: images, themes, storyboards, users and user groups.

2. Once you have retrieved the export file, pass it to the GETIMPORTCONTENT function, which prepares the content that is to be imported into a ContentResource object.
3. You can use then use the details of this object to set another object called **ImportOption**, and pass it to the IMPORTCONTENT or IMPORTCONTENTNOVALIDATION functions to import desired content into Yellowfin. The ImportOption lets you specify which content is to be imported by using the optionKey parameter.
4. Note, however, that if you skip the ImportOption object when importing, the entire contents contained in the file, will be imported.
5. To test or validate the content to be imported, use the TESTIMPORTCONTENT or TESTIMPORTCONTENTNOVALIDATION web services (this does not perform any actual data import).
6. Note:  You can find an example of IMPORTCONTENT function at work with user interface to change ImportOption in the Yellowfin installation folder: **development/examples/webservices/ws_admin_import.jsp.** Simply, place this file into the Y**ellowfin/appserver/webapps/ROOT** folder, and adjust the host, port, admin user details. Then run it from your Internet browser (http://<host>:<port>/ws_admin_import.jsp).

Setting up the **ImportContent** object to specify the content that needs to be imported, requires you to define the **optionKey** parameter by providing its **optionValue**. Here are some combinations that you can use:

| optionKey | optionValue |
|---|---|
| SKIP | "true" or "false". |
| OPTION | "REPLACE" or "ADD". |
| EXISTING | UUID of the existing item in the target Yellowfin instance.<br><br>This workds with optionKey=OPTION and optionValue=REPLACE. |
| PASSWORD1 | Data source password if you want the content to be encrypted during import. |

For instance, if you exported one Yellowfin report with its dependencies (data source, view, category, sub category), your **ContentResource** object (which you get with GETIMPORTCONTENT call) will display the following:

| ResourceId | ResourceType | ResourceName | ResourceUUID |
|---|---|---|---|
| 70031 | DATASOURCE | CSV | null |
| 70209 | VIEW | months | b974f55a-269b-4a4b-b1c6-bf9b968be723 |
| null | RPTCATEGORY | Tutorial | a23c2ec6-a2fa-45c7-b5da-dcf3f02e6633 |
| null | RPTSUBCATEGORY | Reports | 58834ae1-2f65-44c0-b6c3-7c9cd2f91bd5 |
| 70279 | REPORT | My Report | fd3794b3-62c0-4cf8-bac0-755e68d9c41e |

Now you need to define import options for all 5 items. ImportOption's **itemIndex** parameter will correspond to the index number of each of the ContentResource items so that for the ContentResource array example above, the following will be displayed:

| ResourceId | ResourceType | importOption itemIndex |
|---|---|---|
| 70031 | DATASOURCE | 0 |
| 70209 | VIEW | 1 |
| null | RPTCATEGORY | 2 |
| null | RPTSUBCATEGORY | 3 |
| 70279 | REPORT | 4 |

But if you want to replace the report with another existing report in our example, and skip the rest of the content from being imported, here is how you will configure the ImportOption object:

| itemIndex | optionKey | optionValue | Note |
|---|---|---|---|
| 0 | SKIP | true | The data source will be skipped. |
| 1 | SKIP | true | The view will be skipped. |

| | | | |
|---|---|---|---|
| 2 | SKIP | true | The category will be skipped. |
| 3 | SKIP | true | The subcategory will be skipped. |
| 4 | SKIP | false | The report will be imported. |
| 4 | OPTION | REPLACE | The existing report will be replaced with the imported one. |
| 4 | EXISTING | 70287 | This is the report ID of the report existing in the target Yellowfin that will be replaced with the imported report. |

Note, however, that if trying to import content as 'new', you must specify its dependencies in the ImportOption object. Therefore, a new report cannot be imported without importing its required dependencies (that is, view, category, and sub category). (Note: You can choose to REPLACE the dependencies of a report with those existing in the target environment, instead of using the ones retrieved from the export file.) See below for the required dependencies of the main content types:

| Content | Required Dependencies |
|---|---|
| Report | View, category, subcategory |
| View | Data source |
| Dashboard | Category, subcategory |

## Main Import Functions

This function reads a provided YFX or XML file and places specific content from it into the ContentResource object that can be imported.

### 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.<br><br>This account must have the “web services” role enabled, and must belong to the default (ie, primary) org. |
| Password | String | Password of the above account. |
| OrgId | Integer | Default (ie, primary) organization ID within Yellowfin. Always set this to 1. |
| Function | String | Web service function. Set this to "GETIMPORTCONTENT". |
| OrgRef | String | This optional parameter can be used to specify a client org. ID. |
| Parameter | String[] | This array contains strings with details of content that is to be imported from a file. The first string is a byte array of UTF-8 string. The second is the file type, that is "YFX" or "XML". |

### 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:<br><br>• SUCCESS<br>• FAILURE |
| ContentResources | ContentResource[] | Object array containing details of Yellowfin's content to be imported. |

## Instructions

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

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

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

- Specify the file containing data that is to be imported:

```
Path path = Paths.get("/Applications/Yellowfin 7.4/qwerty.yfx");

byte[] data = Files.readAllBytes(path);
byte[] encodeBase64 = java.util.Base64.getEncoder().encode(data);
String f = new String(encodeBase64, "UTF-8");
```

- Provide the extension of this file, ie, either YFX or XML:

```
rsr.setParameters(new String[]{f,"YFX"});
```

- 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 elements: StatusCode and ContentResource. (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_ getimportcontent.jsp.
2. Put the file in the root folder: *Yellowfin/appserver/webapps/ROOT*.
3. Adjust the host, port, and admin user details according to your environment.
4. Run *http://<host>:<port>/ws_ getimportcontent.jsp* from your Internet browser.

```
<%
/*            ws_getimportcontent.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.*" %>
<%@ page import="java.nio.file.Files" %>
<%@ page import="java.nio.file.Paths" %>
<%@ page import="java.nio.file.Path" %>

<%
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");                                // set to the password of the above account
rsr.setOrgId(1);
rsr.setFunction("GETIMPORTCONTENT");

Path path = Paths.get("/Applications/Yellowfin 7.4/qwerty.yfx");                // existing file

byte[] data = Files.readAllBytes(path);
byte[] encodeBase64 = java.util.Base64.getEncoder().encode(data);
String f = new String(encodeBase64, "UTF-8");

rsr.setParameters(new String[]{f,"YFX"});

AdministrationServiceResponse rs = adminService.remoteAdministrationCall(rsr);

if ("SUCCESS".equals(rs.getStatusCode()) ) {
                ContentResource[] cr = rs.getContentResources();
                out.write("<br>Success");
                for (ContentResource o : cr){
                            out.write("<br><br>resourceType: " + o.getResourceType());
                            out.write("<br>resourceCode: " + o.getResourceCode());
                            out.write("<br>resourceName: " + o.getResourceName());
                            out.write("<br>resourceDescription: " + o.getResourceDescription());
                            out.write("<br>resourceOrgId: " + o.getResourceOrgId());
                            out.write("<br>resourceId: " + o.getResourceId());
                            out.write("<br>resourceUUID: " + o.getResourceUUID());
                }
} else {
                out.write("Failure");
                out.write(" Code: " + rs.getErrorCode());
}
%>
```

---

This function imports content from an XML or YFX file into Yellowfin.

### 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.<br><br>This account must have the “web services” role enabled, and must belong to the default (ie, primary) org. |
| Password | String | Password of the above account. |
| OrgId | Integer | Default (ie, primary) organization ID within Yellowfin. Always set this to 1. |
| Function | String | Web service function. Set this to "IMPORTCONTENT". |
| OrgRef | String | This optional parameter can be used to specify a client org. ID. |
| Parameters | String[] | This array contains strings with details of the content to be imported and validated. The first string is a byte array of UTF-8 string. The second is the file type, that is "YFX" or "XML". |
| ImportOptions | ImportOption[] | This optional parameter can be used to define how content should be imported. If not specified, Yellowfin will import all the content as new, exactly how it is contained in the file. |

### 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>
         <!--Optional:-->
         <arg0>
            <loginId>admin@yellowfin.com.au</loginId>
                <password>test</password>
            <orgId>1</orgId>
            <function>IMPORTCONTENT</function>
            <parameters>PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0i ... </parameters>
            <parameters>XML</parameters>
            <importOption>
                                <optionIndex>0</optionIndex>
                            <optionKey>OPTION</optionKey>
                            <optionValue>REPLACE</optionValue>
                                <optionIndex>1</optionIndex>
                            <optionKey>EXISTING</optionKey>
                            <optionValue>70279</optionValue>
             </importOption>
          </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:<br><br>• SUCCESS<br>• FAILURE<br><br>Note: The status corresponds to whether or not the call was performed, not if the actual import was carried out. |

### 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>3c25c8a81c971e26bd23d4ed60194fba</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:

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

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

- Specify the file containing data that is to be imported:

```
Path path = Paths.get("/Applications/Yellowfin 7.4/YFexport.xml");

byte[] data = Files.readAllBytes(path);
byte[] encodeBase64 = java.util.Base64.getEncoder().encode(data);
String f = new String(encodeBase64, "UTF-8");
```

- Provide the extension of this file, ie, either YFX or XML:

```
rsr.setParameters(new String[]{f,"XML"});
```

- Specify how the file content should be imported:

```
ImportOption[] options = new ImportOption[2];
options[0] = new ImportOption();
options[0].setItemIndex(0);
options[0].setOptionKey("OPTION");
options[0].setOptionValue("REPLACE");

options[1] = new ImportOption();
options[1].setItemIndex(0);
options[1].setOptionKey("EXISTING");
options[1].setOptionValue("70279");
```

- Set the import options object into the request:

```
rsr.setImportOptions(options);
```

- 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 elements: StatusCode. (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_ importcontent.jsp.
2. Put the file in the root folder: *Yellowfin/appserver/webapps/ROOT*.
3. Adjust the host, port, and admin user details according to your environment.
4. Run *http://<host>:<port>/ws_ importcontent.jsp* from your Internet browser.

```
<%
/*            ws_importcontent.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.*" %>
<%@ page import="java.nio.file.Files" %>
<%@ page import="java.nio.file.Paths" %>
<%@ page import="java.nio.file.Path" %>
<%
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");                                    // set to the password of the above account
rsr.setOrgId(1);
rsr.setFunction("IMPORTCONTENT");

/*            Yfexport.xml contains just one report with no dependencies
                which can be retrieved using an EXPORTCONTENT web service call and passing a single report id.
                FYI. Latest Yellowfin builds do not allow export Yellowfin content WITHOUT dependencies,
                so ImportOption in this example will not suit any YFX file.
                You need to define proper ImportOption anyway
*/
Path path = Paths.get("/Applications/Yellowfin 7.4/YFexport.xml");

byte[] data = Files.readAllBytes(path);
byte[] encodeBase64 = java.util.Base64.getEncoder().encode(data);
String f = new String(encodeBase64, "UTF-8");

rsr.setParameters(new String[]{f,"XML"});

                ImportOption[] options = new ImportOption[2];
                options[0] = new ImportOption();
                options[0].setItemIndex(0);
                options[0].setOptionKey("OPTION");
                options[0].setOptionValue("REPLACE");

                options[1] = new ImportOption();
                options[1].setItemIndex(0);
                options[1].setOptionKey("EXISTING");

                /*            existing report Id. Can be retrieved from ReportHeader table of Yellowfin
database, ReportId column.
                            keep in mind that the reportId changes each time when a user modifies the report.
                            You can use the GETIDFORUUID call to get the valid reportId value for the report.
                */
                options[1].setOptionValue("70279");

rsr.setImportOptions(options);

AdministrationServiceResponse rs1 = adminService.remoteAdministrationCall(rsr);

                if ("SUCCESS".equals(rs1.getStatusCode()) ) {
                            out.write("<br>Test Import Success");
                }
                else {
                out.write("Failure");
                out.write(" Code: " + rs1.getErrorCode());
                }

%>
```

This web service is the same as the IMPORTCONTENT, with the only difference being that it does not validate the data source.

This function tests and validates potential content imports from a YFX or XML file. This function does not import any actual data.

### 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.<br><br>This account must have the “web services” role enabled, and must belong to the default (ie, primary) org. |
| Password | String | Password of the above account. |
| OrgId | Integer | Default (ie, primary) organization ID within Yellowfin. Always set this to 1. |
| Function | String | Web service function. Set this to "TESTIMPORTCONTENT". |
| OrgRef | String | This optional parameter can be used to specify a client org. ID. |
| Parameter | String[] | This array contains strings with details of the content to be imported and validated. The first string is a byte array of UTF-8 string. The second is the file type, that is "YFX" or "XML". |
| ImportOptions | ImportOption[] | This optional parameter can be used to define how content should be imported. |

### 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>
         <!--Optional:-->
         <arg0>
            <loginId>admin@yellowfin.com.au</loginId>
                <password>test</password>
            <orgId>1</orgId>
            <function>TESTIMPORTCONTENT</function>
            <parameters>PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0i ... </parameters>
            <parameters>XML</parameters>
            <importOption>
                                <optionIndex>0</optionIndex>
                            <optionKey>OPTION</optionKey>
                            <optionValue>REPLACE</optionValue>
                                <optionIndex>1</optionIndex>
                            <optionKey>EXISTING</optionKey>
                            <optionValue>70279</optionValue>
             </importOption>
          </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:<br>• SUCCESS<br>• FAILURE |
| ImportIssues | ImportIssue[] | Object array containing issues faced while importing the file. |
| ContentResources | ContentResource[] | Object array containing details of Yellowfin's content to be imported. |

### 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>
            <contentResources>
               <resourceDescription>months, 6/3/2018 8:53 AM</resourceDescription>
               <resourceId>70279</resourceId>
                   <resourceName>My Report 1000</resourceName>
                   <resourceType>REPORT</resourceType>
               <resourceUUID>fd3794b3-62c0-4cf8-bac0-755e68d9c41e</resourceUUID>
            </contentResources>
            <errorCode>0</errorCode>
                <importIssues>
                   <issueElements>
                      <messageKey>error.reports.import.view</messageKey>
                      <renderedMessage>View has not been selected.</renderedMessage>
                   </issueElements>
                   <issueElements>
                  <messageKey>error.reports.import.category</messageKey>
                      <renderedMessage>Folder has not been selected.</renderedMessage>
                   </issueElements>
                   <resource>
                  <resourceDescription>months, 6/3/2018 8:53 AM</resourceDescription>
                      <resourceId>70279</resourceId>
                      <resourceName>My Report 1000</resourceName>
                  <resourceType>REPORT</resourceType>
                  <resourceUUID>fd3794b3-62c0-4cf8-bac0-755e68d9c41e</resourceUUID>
                   </resource>
            </importIssues>
            <messages>Successfully Authenticated User: admin@yellowfin.com.au</messages>
            <messages>Web Service Request Complete</messages>
            <sessionId>ab398569ce36672e9d776c3dae3804d6</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:

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

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

- Specify the file containing data that is to be imported:

```
Path path = Paths.get("/Applications/Yellowfin 7.4/qwerty.yfx");

byte[] data = Files.readAllBytes(path);
byte[] encodeBase64 = java.util.Base64.getEncoder().encode(data);
String f = new String(encodeBase64, "UTF-8");
```

- Provide the extension of this file, ie, either YFX or XML:

```
rsr.setParameters(new String[]{f,"YFX"});
```

- 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.

- Then test the imported content:

```
if ("SUCCESS".equals(rs.getStatusCode()) ) {
                out.write("<br>Success");

                ContentResource[] crs = rs.getContentResources();

                ImportIssue[] ImportIssues = rs.getImportIssues();
                out.write("<br>Import Issues: " + (ImportIssues!=null?ImportIssues.length:"no issues"));

                out.write("<table>");
                out.write("<tr><td> ResourceId </td><td> ResourceType </td><td> ResourceName </td><td>
ResourceUUID </td></tr>");
                for (ContentResource c: crs) {
                            out.write("<tr>");
                            out.write("<td>" + c.getResourceId() + "</td><td>" + c.getResourceType() + "<
/td><td>" + c.getResourceName() + "</td><td>" + c.getResourceUUID() + "</td>");
                            out.write("</tr>");
   }
} else {
                out.write("Failure");
                out.write(" Code: " + rs.getErrorCode());
}
```

- The response will contain the following elements: StatusCode, ImportIssues and ContentResources. (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_ testimportcontent.jsp.
2. Put the file in the root folder: *Yellowfin/appserver/webapps/ROOT*.
3. Adjust the host, port, and admin user details according to your environment.
4. Run *http://<host>:<port>/ws_ testimportcontent.jsp* from your Internet browser.

```
<%
/*            ws_testimportcontent.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.*" %>
<%@ page import="java.nio.file.Files" %>
<%@ page import="java.nio.file.Paths" %>
<%@ page import="java.nio.file.Path" %>
<%
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");                                       // set to the password of the above account
rsr.setOrgId(1);
rsr.setFunction("TESTIMPORTCONTENT");

Path path = Paths.get("/Applications/Yellowfin 7.4/www.yfx");                        // existing file

byte[] data = Files.readAllBytes(path);
byte[] encodeBase64 = java.util.Base64.getEncoder().encode(data);
String f = new String(encodeBase64, "UTF-8");

rsr.setParameters(new String[]{f,"YFX"});

AdministrationServiceResponse rs = adminService.remoteAdministrationCall(rsr);

if ("SUCCESS".equals(rs.getStatusCode()) ) {
                out.write("<br>Success");

                ContentResource[] crs = rs.getContentResources();

                ImportIssue[] ImportIssues = rs.getImportIssues();
                out.write("<br>Import Issues: " + (ImportIssues!=null?ImportIssues.length:"no issues"));

                out.write("<table>");
                out.write("<tr><td> ResourceId </td><td> ResourceType </td><td> ResourceName </td><td>
ResourceUUID </td></tr>");
                for (ContentResource c: crs) {
                            out.write("<tr>");
                            out.write("<td>" + c.getResourceId() + "</td><td>" + c.getResourceType() + "<
/td><td>" + c.getResourceName() + "</td><td>" + c.getResourceUUID() + "</td>");
                            out.write("</tr>");
   }
} else {
                out.write("Failure");
                out.write(" Code: " + rs.getErrorCode());
}
%>
```

---

This web service performs the same content import tests as TESTIMPORTCONTENT, but unlike TESTIMPORTCONTENT,  it does not validate the data source.

## Content Translation Functions

Yellowfin's Content Translation functionality allows users to translate content, such as reports, views, dashboards, etc. from a previously configured language. Click here to learn more about this feature and the translation process involved. The following web services involve exporting or importing translated content.

**Note:** Ensure that you have defined other languages in Yellowfin before exporting translatable content.

This web service exports translated content into a CSV file. The data returned is translatable content across all active views, reports, and dashboards.

### 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.<br><br>This account must have the “web services” role enabled, and must belong to the default (ie, primary) org. |
| Password | String | Password of the above account. |
| OrgId | Integer | Default (ie, primary) organization ID within Yellowfin. Always set this to 1. |
| Function | String | Web service function. Set this to "EXPORTTRANSLATIONALL". |

### 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>EXPORTTRANSLATIONALL</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:<br>• SUCCESS<br>• FAILURE |
| ContentType | String | Type of the export file. For example, text/comma-separated-values |
| FileName | String | Generated file name. |
| BinaryData | String | Base64 encoded string with translated data. |

### 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>
            <binaryData>77u/VVVJRCxUZXh0IFR5cGUsS2V5LE9yaWdpbmFsIFRleHQs ... </binaryData>
            <contentType>text/comma-separated-values</contentType>
            <errorCode>0</errorCode>
            <fileName>Translations - 10 Mar 2018.csv</fileName>
            <messages>Successfully Authenticated User: admin@yellowfin.com.au</messages>
            <messages>Web Service Request Complete</messages>
            <sessionId>4a19aa468b23ab18d3aee5c7121bcacd</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:

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

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

- 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 elements: StatusCode, BinaryData, FileName and ContentType. (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_ exporttranslationall.jsp.
2. Put the file in the root folder: *Yellowfin/appserver/webapps/ROOT*.
3. Adjust the host, port, and admin user details according to your environment.
4. Run *http://<host>:<port>/ws_ exporttranslationall.jsp* from your Internet browser.

```
<%
/*            ws_exporttranslationall.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.*" %>
<%@ page import="java.nio.file.Files" %>
<%@ page import="java.io.PrintWriter" %>
<%
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");                            // set to the password of the above account
rsr.setOrgId(1);
rsr.setFunction("EXPORTTRANSLATIONALL");

AdministrationServiceResponse rs = adminService.remoteAdministrationCall(rsr);

                if ("SUCCESS".equals(rs.getStatusCode()) ) {
                            out.write("<br>Success");

                            //response.setBinaryData(Base64.encodeBytes(pdf.getData()));

                            String Base64encoded = rs.getBinaryData();
                            Base64encoded = Base64encoded.replace("\n", "").replace("\r", "");

                            byte[] bytes = Base64encoded.getBytes();
                            byte[] decoded = java.util.Base64.getDecoder().decode(bytes);

                            String text = new String(decoded, "UTF-8");
                            PrintWriter writer = new PrintWriter("/Applications/Yellowfin 7.4/" + rs.
getFileName(), "UTF-8");
                            writer.println(text);
                            writer.close();
                }
                else {
                out.write("<br>Failure");
                out.write(" Code: " + rs.getErrorCode());
                }
%>
```

---

This function imports a translation CSV file into Yellowfin. This file is generated during the content translation process when specified content is export. Along with other details, the file contains columns for other specified languages. Add the translated content in their designated column and use this function to import the file back into the system.

### 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.<br><br>This account must have the “web services” role enabled, and must belong to the default (ie, primary) org. |
| Password | String | Password of the above account. |
| OrgId | Integer | Default (ie, primary) organization ID within Yellowfin. Always set this to 1. |
| Function | String | Web service function. Set this to "IMPORTTRANSLATION". |
| BinaryData | String[] | Array containing bytes that represent a translation CSV file. |

### 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>IMPORTTRANSLATION</function>
                        <binaryData>-17</binaryData>
                        <binaryData>-69</binaryData>
                        <binaryData>-65</binaryData>
                        <binaryData>85</binaryData>
                        <binaryData>85</binaryData>
                                ...
                </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:<br><br>• SUCCESS<br>• FAILURE<br><br>The status relates to whether the file import was successfull or not. |

### 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>11c466874bfbcdb80f5d250c97ffbd03</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:

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

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

- Specify the translation CSV file to be imported:

  ```
  Path path = Paths.get("/Applications/Yellowfin 7.4/Translations - 8 Mar 2018.csv"); // existing file
  byte[] data = Files.readAllBytes(path);

  rsr.setBinaryData(data);
  ```

- 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 elements: StatusCode. (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_ importtranslation.jsp.
2. Put the file in the root folder: *Yellowfin/appserver/webapps/ROOT*.
3. Adjust the host, port, and admin user details according to your environment.
4. Run *http://<host>:<port>/ws_ importtranslation.jsp* from your Internet browser.

```
<%
/*            ws_importtranslation.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.*" %>
<%@ page import="java.nio.file.Files" %>
<%@ page import="java.nio.file.Paths" %>
<%@ page import="java.nio.file.Path" %>
<%

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");                                // set the password of the account above
rsr.setOrgId(1);
rsr.setFunction("IMPORTTRANSLATION");

Path path = Paths.get("/Applications/Yellowfin 7.4/Translations - 8 Mar 2018.csv");                // existing
file

byte[] data = Files.readAllBytes(path);

rsr.setBinaryData(data);

AdministrationServiceResponse rs = adminService.remoteAdministrationCall(rsr);
                if ("SUCCESS".equals(rs.getStatusCode()) ) {
                            out.write("<br>Success");
                }
                else {
                out.write("<br>Failure");
                out.write(" Code: " + rs.getErrorCode());
                }
%>
```