Talent Analytics API - Documentation

Version 2012-01-01	

Overview

Talent Analytics' web service can be accessed using SOAP web services. This interface is described by a Web Services Description Language (WSDL) document which defines the operations, data types and security model. For more information on WSDL and SOAP, please see refer to the W3C Specification

Versioning

All schemas have a version number. The version number appears in the URL of a schema file, and in a schema's target namespace. The latest version is 2012-01-01. Upgrading is made easy by differentiating requests based on the version number. In addition to the latest version, the service will support the older versions for some time. Once customer transition to the new version is complete, the older versions will be retired.

The Talent Analytics API WSDL can be found at URLs of the form 'https://api.talentanalytics.com/doc/VERSION/taapi.wsdl' where VERSION indicates the version of the API. The current API version is 2012-01-01 and can thus be found at URL https://api.talentanalytics.com/doc/2012-01-011/taapi.wsdl.

Authentication

Authentication is handled by supplying an "AuthHeader" node in the soap header. Implementation can vary from client to client. The added header should look like this:

```
<AuthHeader>
  <Username>ponzi@talentanalytics.com</Username>
  <Password>m0@rCh33z3</Password>
</AuthHeader>
```

Creating a Respondent

When a Respondent is created using a valid email address, Talent Analytics will return a RespondentInfo object. This object will include the status of that

Respondent (which will always be 'pending' for newly created Respondents). It will also contain a RespondentID, which is TalentAnalytics unique identifier. You can use this later when making queries about the Respondent. In Addition to EMail we recommend, but do not require, sending: FirstName,LastName, and Gender. Gender will be used to improve the natural feel of the content we create.

extID

You may optionally specify your own External ID (ExtID) for similar look up purposes. This ID should, but does not necessarily have to be unique. If you don't already have your own outward-facing Identifier for this Employee, we suggest using an Employee ID number combined with something such as a Company ID, so that it is unique to your system.

callbackURL

When calling createRespondent, you may specify an optional "callbackURL". Talent Analytics' API servers will make a GET request to this URL when important information about a Respondent has changed. This URL should be unique to your employee. Since the request is made without any POST information, we suggest using some sort of key in theURL. An example would be:

```
http://analytics.example.com/employeeUpdate/xxxxxxxxxx/?eid=12345
```

Where xxxxxxxxxx is a hash that prevents a rogue automated process from making troublesome requests. In this case, eid is your own identifier, which will allow you to know which respondent we are updating you about. As of now, the only event that triggers a request to this URL is the successful completion of a questionnaire by the Respondent. This notification will allow you to make a call to getRespondentInfo with the RespondentID to see what has changed. If the respondent is no longer "pending" (i.e "complete") you can make further requests via getContent.

This URL can be anything that is useful to you. We ignore the response. If we receive an HTTP error, our servers will make a second attempt within the next few minutes. A second failure will prompt us to stop making requests regarding the particular event that triggered it. As of now, no alerts are sent in this case. It is a good idea to periodically query our API on any "pending" respondents.

Finding Respondents, Getting and Updating Respondent Info

Talent Analytics' API provides a small handful of methods to manage Respondent information. All of these use a similar "RespondentInfo" object.

findRespondents

When a query is made to findRespondent, you will provide a RespondentInfo object containing any information you know about a respondent. This method is only necessary if you do not already have their RespondentID. It will return a list of RespondentInfo objects. One for each Respondent found.

This method only returns results where all supplied fields at least partially match. If you supply a RespondentInfo object such as:

```
{EMail:“@dundermifflin.com”, Gender:"M"}
```

… you will receive a list of all respondents where EMail address contains “@dundermifflin.com” AND Gender matches “M”. The return format is a list of RespondentInfo objects containing basic info about each Respondent (FirstName, LastName, Gender, ExtID, Status, RespondentID). You can then use the RespondentIDs to make in-depth queries using getRespondentInfo or getContent.

getRespondentInfo

Given a list of RespondentIDs, a call to getRespondentInfo will return a RespondentList. This is the preferred method if you already know the RespondentID for whomever you would like information on. By default, the returned RespondentInfo objects will be populated similarly to the ones returned for findRespondents. You may, however, supply an optional list of Fields you would like returned, including additional information such as CORE and Ambitions scores (if licensed) and any Labels applied to each Respondent.

updateRespondentInfo

Given a single RespondentID, and a RespondentInfo object, a call to updateRespondentInfo allows you to change any editable Respondent data. Only supplied values will be changed. Allowed changes include FirstName, LastName, Email, Gender, CallbackURL, ExtID.

Labels

Labels can be used to organize Respondents in our System. Talent Analytics’ API provides 3 methods for managing Respondent Labels. If two API accounts have access to the same Respondent, they each have their own set of labels for a given respondent.

For API methods that take, as an argument a list of one or more RespondentIDs, a Label can be used by appending a hash in front of the string. For instance, if you

want to call “getRespondentInfo” on all Respondents who have been Labled as “sales-rep”, make the following call:

```
getRespondentInfo([ “#sales-rep” ])
```

Further, you can mix Labels with ordinary RespondentIDs as such:

```
getContent([ “#sr-dev”, “YVN40T5210”, “GBK93E8443”], “tips_for_communicating”)
```

This will return “Tips for Communicating” content for any respondent labeled as “sr-dev” as well as the two Respondents identified by the remaining ids.

setRespondentLabels

Applies the supplied list of labels to each respondent in the given list. Do not use the hash mark here.

removeRespondentLabels

Removes the supplied list of labels from each respondent in the given list. Do not use the hash mark here.

getRespondentLabels

Returns a list of labels for for the given RespondentID as set by the user.
Note that the same list of labels can be obtained by calling getRespondentInfo and adding “Labels” to the optional FieldList parameter

getContent

Most licensed content will be obtained with a call to getContent. This method Returns the content for a each respondent given a list and content a content key. An optional format descriptor (html by default) can be supplied to specify the output type when applicable. Any HTML content returned will be translated
using XML character entity references (i.e >).

METHODS

Method	Input	Output	Description
createRespondent	RespondentInfoType	RespondentInfoType	Creates an RDC request for the specified user, email address is required as part of the given respondent

			info input. All other supplied info is added at creation time. Returns the respondent info.
getRespondentInfo	ObjectIDListType,FieldListType	RespondentInfoListType	Returns the respondent info for the given id. Restricts output only to supplied fields. If no fields are supplied, returns the entire info object.
updateRespondentInfo	ObjectIDType,RespondentInfoType	boolean	Updates a single respondent info for the given id. Only overwrites fields that are present in the passed in respondent info package.
setRespondentLabels	ObjectIDListType,LabelListType	boolean	Applies the supplied list of labels to each respondent in the given list.
removeRespondentLabels	ObjectIDListType,LabelListType	boolean	Removes the supplied list of labels from each respondent in the given list.
getRespondentLabels	ObjectIDType	LabelListType	Returns a list of labels for for the given id as set by the user.
getContent	ObjectIDListType,ContentKeyType,ContentFormatType	RespondentContentListType	Returns the content for a list of respondents given a list and content key. Takes an optional format descriptor (html by default)
findRespondents	RespondentInfoType	RespondentInfoListType	Does a search for respondents based on supplied info.

Simple Types

Field	Type
ObjectIDType	string
ContentKeyType	string
ContentFormatType	string
GenderType	string
ObjectIDListType	array
RespondentInfoListType	array
RespondentContentListType	array

LabelListType	array
FieldListType	array

Complex Types

RespondentInfoType (standard)

Field	Type	Description
RespondentID	ObjectIDType	Unique Identifier assigned by Talent Analytics
ExtID	string	User-specified Identifier
CallbackURL	string	User-specified URL, will called by Talent Analytics when there are important updates for this Respondent.
FirstName	string	
LastName	string	
EMail	string	
Gender	GenderType	Gender of Respondent: [M,F,NA]
Status	string	Status of Respondent: [pending,complete,inactive]
Labels	LabelListType	Labels, if any, associated with this respondent.

Additional RespondentInfoType Fields (if licensed)

Field	Type	Description
MATCH_TOTAL	double	Total Match to default Benchmark
MATCH_CORE	double	Match to default CORE Benchmark
MATCH_AMB	double	Match to default Ambitions Benchmark
Field	**Type**	**Description**
ACT_C	double	
ACT_O	double	
ACT_R	double	
ACT_E	double	
Field	**Type**	**Description**
MOD_C	double	
MOD_O	double	
MOD_R	double	

MOD_E	double	

Field	Type	Description
SCORE_ECO	double	
SCORE_ALT	double	
SCORE_THE	double	
SCORE_AUT	double	
SCORE_POL	double	
SCORE_IND	double	
SCORE_CRE	double	

Field	Type	Description
RANK_ECO	double	
RANK_ALT	double	
RANK_THE	double	
RANK_AUT	double	
RANK_POL	double	
RANK_IND	double	
RANK_CRE	double	

RespondentContentType

Field	Type	Description
RespondentID	ObjectIDType	ID of the respondent of whom this content describes.
ContentKey	ContentKeyType	A key that classifies the content. Typically this and RespondentID are a unique pair and can be considered a foreign key to the content.
Format	ContentFormatType	Format of the content found in the Text field (html,textism,raw)
Text	string	The actual content

Content Available via getRespondentContent and the associated keys:

ContentKey	Description
overview	Executive Overview
strengths	Primary Strengths

work_environment	Optimal Work Environment
tips_for_communicating	Tips for Communicating
learning_environment	Ideal Learning Environment
performance_flags	Performance Flags
tips_for_managing	Tips for Managing
tips_for_motivating	Tips for Motivating

Sample Request XML

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns1="https://a
pi.talentanalytics.com/2012-01-01" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://ww
w.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema
-instance"SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Header>
  <AuthHeader>
   <Username>ponzi@talentanalytics.com</Username>
   <Password>m0@rCh33z3</Password>
  </AuthHeader>
 </SOAP-ENV:Header>
 <SOAP-ENV:Body>
  <ns1:getRespondentInfo>
   <RespondentIDList SOAP-
ENC:arrayType="ns1:ObjectIDType[3]" xsi:type="ns1:ObjectIDListType">
    <item xsi:type="ns1:ObjectIDType">BRC81TGTYS7729</item>
    <item xsi:type="ns1:ObjectIDType">BRC81TGTYS7730</item>
    <item xsi:type="ns1:ObjectIDType">BRC92TTYGD2955</item>
   </RespondentIDList>
   <FieldList SOAP-
ENC:arrayType="xsd:string[3]" xsi:type="ns1:FieldListType">
    <item xsi:type="xsd:string">FirstName</item>
    <item xsi:type="xsd:string">LastName</item>
    <item xsi:type="xsd:string">EMail</item>
   </FieldList>
  </ns1:getRespondentInfo>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Sample Response XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:
SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"xmlns:xsd="http://www
.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="https://a
pi.talentanalytics.com/2012-01-01">
 <SOAP-ENV:Body>
  <ns1:getRespondentInfoResponse xmlns:ns1="https://api.talentanalyt
ics.com/2012-01-01">
   <RespondentInfoList xsi:type="SOAP-ENC:Array" SOAP-
ENC:arrayType="tns:RespondentInfoType[3]">
    <item xsi:type="tns:RespondentInfoType">
     <RespondentID xsi:type="tns:ObjectIDType">BRC81TGTYS7729</Respo
ndentID>
     <FirstName xsi:type="xsd:string">Finn</FirstName>
     <LastName xsi:type="xsd:string">Berry</LastName>
     <EMail xsi:type="xsd:string">finn@talentanalytics.com</EMail>
    </item>
    <item xsi:type="tns:RespondentInfoType">
     <RespondentID xsi:type="tns:ObjectIDType">BRC81TGTYS7730</Respo
ndentID>
     <FirstName xsi:type="xsd:string">Finn</FirstName>
     <LastName xsi:type="xsd:string">Karofsky</LastName>
     <EMail xsi:type="xsd:string">finn@talentanalytics.com</EMail>
    </item>
    <item xsi:type="tns:RespondentInfoType">
     <RespondentID xsi:type="tns:ObjectIDType">BRC92TTYGD2955</Respo
ndentID>
     <FirstName xsi:type="xsd:string">Will</FirstName>
     <LastName xsi:type="xsd:string">Berry</LastName>
     <EMail xsi:type="xsd:string">will@talentanalytics.com</EMail>
    </item>
   </RespondentInfoList>
  </ns1:getRespondentInfoResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Sample Code

PHP

getRespondentInfo

The following code generates a request similar to the SOAP sample above, and also parses the response. It requests RespondentInfo based on 3 RespondentIDs, limiting the response to a handful of fields. It then displays the name of each Respondent.

```
<?php
$wsdlurl = "http://api.talentanalytics.com/doc/2012-01-
01/taapi.wsdl";

$trace = 1;


$client = new SoapClient($wsdlurl,array('trace'=>$trace));

//create an auth header.
$str_header_component= "<AuthHeader><Username>something</Username><P
assword>foo</Password></AuthHeader>";
$obj_var_auth_inside = new SoapVar($str_header_component, XSD_ANYXML
, null, null, null);
$obj_header_auth_outside = new SoapHeader('https://api.talentanalyti
cs.com/2012-01-01', 'AuthHeader', $obj_var_auth_inside);
$client->__setSoapHeaders(array($obj_header_auth_outside));

$returned_data = null;

try {
  $respondent_id_list = array('BRC81TGTYS7729','BRC81TGTYS7730','BRC
92TTYGD2955');
  $returned_data =  $client-
>getRespondentInfo($respondent_id_list,array('FirstName','LastName',
'EMail'));
}
catch(Exception $e) {
  print 'something bad happened';
}

if(!empty($returned_data)) {
  foreach($returned_data as $respondent_info) {
        echo $respondent_info->FirstName . ' ' .  $respondent_info-
>LastName . "\n";
  }
}
```

createRespondent

The following code takes an employee from a client database, and makes a Rapid Data Collection request to Talent Analytics.

```
<?php
$wsdlurl = "http://api.talentanalytics.com/doc/2012-01-
01/taapi.wsdl";

$api_user = 'taapiuser@example.com';
$api_pass = 'supersecure'; //you didn't actually store this in your
script, did you?!?

//create an auth header.
$str_header_component= "<AuthHeader><Username>$api_user</Username><P
assword>$api_pass</Password></AuthHeader>";
$obj_var_auth_inside     = new SoapVar($str_header_component, XSD_AN
YXML, null, null, null);
$obj_header_auth_outside = new SoapHeader('https://api.talentanalyti
cs.com/2012-01-01', 'AuthHeader', $obj_var_auth_inside);

$client = new SoapClient($wsdlurl);
$client->__setSoapHeaders(array($obj_header_auth_outside));

$my_employee = array(

    /* Required Fields */
        'FirstName'   => 'Michael',
        'LastName'    => 'Scott',
        'EMail'       => 'michael.scott@dundermifflin.com',

    /* Optional Fields */
        'Gender'      => 'M',        //defaults to NA
        'ExtID'       => 'scra-011', //often employee ID, but should
be unique to client
        'CallbackURL' => 'https://internal.dundermifflin.com/emp-
update/?emp_id=scra-011&key=7Kd9zz9Df',

        'Labels'      => array('managers')

);

$soap_reply = null;
try {
  $soap_reply =  $client-
```

```
>createRespondent(new SoapVar($my_employee,SOAP_ENC_OBJECT));
}
catch(SoapFault $e) {
  echo "Something bad happened:\n";
  echo $e->faultcode   . "\n";
  echo $e->faultstring . "\n";
}

if(null !== $soap_reply) {

  //Talent Analytics assigns a RespondentID to your Employee, which
you can use for future lookups
  $remote_id = $soap_reply->RespondentID;
        /*
        results:
  object(stdClass)#5 (6) {
  ["RespondentID"]=>
        string(10) "8GF1L29865"
  ["ExtID"]=>
        string(8) "scra-011"
  ["FirstName"]=>
        string(7) "Michael"
  ["LastName"]=>
        string(5) "Scott"
  ["Gender"]=>
        string(1) "M"
  ["Status"]=>
        string(7) "pending"
        }
        */
}
```

Python

getRespondentInfo

The following code generates a request for a single respondent using getRespondentInfo

```
import suds
from suds.sax.element import Element

url = "https://api.talentanalytics.com/doc/2012-01-01/taapi.wsdl"
```

```
client = suds.client.Client(url)

authheader = Element("AuthHeader")
un = Element("Username").setText("someuser@example.com")
pw = Element("Password").setText("supersecure")
authheader.append(un).append(pw)
client.set_options(soapheaders=authheader)

try:
        result = client.service.getRespondentInfo(["YVN40T5210"])
        print result
except suds.WebFault as detail:
        print detail.fault.faultcode
        print detail.fault.faultstring
```

Javascript (google apps flavor)

This example shows how to get a list of CORE scores given a respondent id using a typical SOAP client. In this case, google app script.

```
var config = {
    username : "taapiuser@example.com",
    password : "supersecret",
    wsdlurl  : "http://api.talentanalytics.com/doc/2012-01-01/dev-
taapi.wsdl"
};

var respondentID = "YVN40T5210";

var wsdl    = SoapService.wsdl(config.wsdlurl);
var taxml   = wsdl.getService("TAAPI");
var header  = Xml.element("AuthHeader", [Xml.element("Username",[con
fig.username]),Xml.element("Password",[config.password])]);
var res     = taxml.getRespondentInfo(["item",respondentID],"",heade
r);
var retNode = res.Envelope.Body.getRespondentInfoResponse.Respondent
InfoList.item;

//array to store the scores
var scores = [];

score_types = ["ACT_C","ACT_O","ACT_R","ACT_E"];
```

```
for(var i = 0;i<score_types.length;i++) {
    theScore = parseFloat(retNode[score_types[i]].getText());
    scores[score_types[i]] = theScore;
}
```