ORACLE

TALEO
CLOUD SERVICE

TALEO BUSINESS EDITION
REST API REFERENCE GUIDE

AUGUST 15, 2020

Part Number: E57841-03

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Hardware and Software, Engineered to Work Together

CONTENTS

ADVISORY

Please ensure that you are working with the latest version of the TBE REST API GUIDE.

The latest version is available on the Oracle Technology Network at https://www.oracle.com/technical-resources/documentation/taleobusiness.html

WELCOME

Information Confidentiality

The Taleo Business Edition REST API and its associated documents, including this guide, are provided for Taleo customers and partners solely. This document should not be viewed, reviewed, replicated, referred to, or reproduced by individuals not a current a Taleo Business Edition customer and/or partner.

It shall be agreed by the recipient of the document (hereafter referred to as "the other party") that confidential information disclosed by Taleo through its documents shall be retained in confidence by the other party, and its respective employees, affiliates and/or subsidiaries, pursuant to the following terms and conditions:

1. Any information, know-how, data, process, technique, design, drawing, program, formula or test data, work in process, business plan, sales, suppliers, customer, employee, investor or business information contained in a document, whether in written, graphic, or electronic form; or
2. Any document, diagram, or drawing which is either conspicuously marked as "Confidential", known or reasonably known by the other party to be confidential, or is of a proprietary nature, and is learned or disclosed in the course of discussions, demonstrations, or other collaboration undertaken between the parties.

Audience and Background

This guide is intended for developers who plan to utilize the Taleo Business Edition for integration with third-party products and services.

The Taleo Business Edition REST API provides programmatic access to the Taleo Business Edition application. Developers are able to build custom applications and services in their platform of choice that supports the open standard REST "Web Services" methodology.

Required Knowledge and Skills

Use of this guide assumes you are already familiar with the following:

- Taleo Business Edition (click the Help section from your instance)
- HTML (details at https://www.w3schools.com/html/default.asp)
- XML and XHTML (details at https://www.w3schools.com/xml/default.asp)
- Web Services (details at https://www.w3schools.com/webservices/default.asp)

A REST client can be written in a wide range of development tools and platforms. As such, Taleo does not provide platform code support or sample code to developers. The API is inherent to the application, not a supported module, and Taleo assumes that developers are experts in their platform of choice with utilizing web services.

Support

Contact Oracle Global Customer Support for any technical issues as they pertain to the API or feedback regarding this documentation. You may contact Customer Support at https://support.oracle.com. If you have purchased support in the US you may also call 1.800.223.1711.

QUICK START NOTES

Authentication	Cookie based authentication: This method requires submission of username, password and company code credentials as part of login resource request. The returning session value provided has to be set in subsequent calls as part of the cookie within the header. Configured as: authToken=<<webapi_token>>.
HTTPS / SSL	We require that all requests are done over SSL.
Versioning	Taleo Business Edition API has not started versioning; however this attribute has been included in the URL structure for potential usage at a later time. If this does occur, Taleo will communicate the potential changes to your Taleo Administrator. Taleo current defaulted version of the API is "1" (i.e. …/ats/api/**v1**/object).
UTF-8 Encoding	Every string passed to and from the Taleo Business Edition API needs to be UTF-8 encoded.
Date Format	All dates in the API are strings in the following format: "YYYY-MM-DD"**
Locale Global Parameter	Taleo Business Edition uses the locale parameter of your company specifications for content responses. This is at the global setting stage that was created when you first implemented Taleo Business Edition. This is not user specific.
JSON vs XML	All URL requests are expected JSON format and responses will follow suit. When adding attribute of ".xml" to the request URL, the format expected on the request will be XML and response will default XML.
URL Structure	…/object/<resource>?<parameter>&<parameter> for JSON, where parameters will be provided throughout this document. For XML, append .xml to the url request: …/object/<resource>.xml?<parameter>&<parameter>.
Discovery of Host (EndPoint)	Taleo Business Edition customers can be accessed via different endpoint URL's. These are described as a <<HOST_URL>> in this document. To find a customer's endpoint, you will need to use the dispatcher service. Please note that customer's can be moved at any given time, as such continuous querying for a customer's endpoint is recommended.
Discovery of Objects	Taleo Business Edition API provides a discovery call enabling programmatic view at APIs can be accessed. The getObject call provides a view into the entire API's available per customer's Taleo instance (for classic REST API's: <<HOST_URL>>/object/info; for JSON Hyperschemas: <<HOST_URL>>/object/info/hyperschema).
Discovery of MetaData	Taleo Business Edition API provides a discovery call enabling programmatic view at what fields can be accessed. The getMetadata call provides a view into the specific fields available for a given object. Two types of MetaData are available. To view the TBE classic REST API MetaData: (<<HOST_URL>>/object/<<OBJECT_LABEL_NAME>>/description/custom). To view the TBE JSON Hyperschema MetaData: (<<HOST_URL>>/object/<<OBJECT_LABEL_NAME>>/hyperschema.json). Note the TBE JSON Hyperschemas are not available for Activity, CompanyGoals, Displayfield, EmployeeGoal, HistoryLog, ResumeToCandidate, Transaction, and ParseResume.

Discovery of Fields	Taleo Business Edition API provides a discovery call enabling programmatic view at field definitions in Taleo Business Edition. The get and put displayFields calls will support this (<<HOST_URL>>/object/displayfield/<<OBJECT_ENTITY_CODE>>/<<OBJECT_FIELD_EXTERNAL-API_NAME>>).

INTRODUCTION – WHY REST

Taleo Business Edition includes programmatic information access through a Representational State Transfer (REST) API. The REST API allows *existing* Taleo system users (with a user role of **Administrator**) to login to the API through built development. Once logged in, users will have full Administrator access to the environment.xc

For those familiar with Taleo Business Edition would already know about the Taleo Business Edition SOAP API and open integration approach methodology. The REST API is provided in the same method, all Premium Support level customers have access to the integration framework at no additional cost or configuration. The only difference is utilization of REST technology vs SOAP. So why the new framework?

REST technology was originally built for Taleo to build stronger and more flexible integrations for social media, mobile, OEM applications, and 3rd party solutions for customers (via Taleo Connect). We have opened up the framework for usage by Taleo partners (refer to Solutions Exchange) and customers now so they can also benefit from the technology set. Here are just a few benefits of REST (vs our legacy SOAP API):

- Generally faster integration build-times for development staff
- Browser based supported integrations (i.e. mobile support)
- Better structure, data and uniformity in requests and responses
- Easier navigation to subset data (related data links in response)
- Better scalability with statelessness
- Single field-level updates to records (vs having to send entire record set back)
- Customizable filtered result sets
- Updating Field Level Picklists (Adding & Deactivating)
- Extended Employee and Administration API support
- Batch search responses
- Two language points (XML and JSON)
- Security Manager support (Encrypted data user rule support)
- No requirement to manage WSDL versions
- Support of Taleo Business Edition PERFORM and Taleo ONBOARD

INTRODUCTION – REST FRAMEWORK

The REST API provided is in a passive manner. Passive meaning the API sits at rest which you can query as often or as little as required through your developer tools. Your built integration would login to Taleo programmatically, query for the prevalent data it is looking for (i.e. find new hires over past 24 hours), extract the data in Taleo JSON/XML format, reformat to what's required for your 3rd party system, and then send it through to that system. Once you have completed engagement through Taleo's API for that session, please logout of the service for security purposes.

Taleo provides the application programming interface infrastructure at no cost for customers on the Premium Service Level. This documentation provides you getting started instructions, along with JSON and XML examples of API requests and responses. Taleo does not provide sample developer code or support outside of raw Taleo JSON/XML, nor does Taleo validate API interactions with 3rd party systems if you have chosen to build integrations on your own. Please ensure your development team has experience with building web service integrations and in the platform of their choice. If you are interested in a hands-on integration built and/or supported by Taleo, please contact Taleo for the appropriate scoping and package purchase.

REST RESOURCES

The Taleo Business Edition REST API focuses on resources of a specific Taleo Business Edition instance. A resource is any layer of information in context. This can include object records like an employee record, a

location, a region, a work history instance, etc. Or can include discovery level data like licensed services, metadata or display fields.

METHODS

Taleo Business Edition uses the standard GET, PUT, POST and DELETE methods for the REST API. Each method defines the type of request to a resource; where that resource can be discover data, object record(s) or specialty functions supported within the application (i.e. upsert data, parse data). The following lists the HTTP methods Taleo Business Edition supports:

Method	Description
GET	• Retrieval of a resource; Either a **Get**, **Get All** or **Search**: ○ Object record through direct link. ○ Discovery data through direct link. ○ Object record(s) through search parameters (where Search is supported in application). ○ Object records through Get All function (where Search is not-supported in application).
PUT	• Update of resource through provided representation; Or **Update**: ○ Object record with data. ○ Look Up values in a picklist for a given display field.
POST	• Creation of resource through provided representation; Or **Create**: ○ Object record with data*. ○ Session Login key (cookie). ** Where Upsert parameter of an object acts like a PUT, will update if matching record found or create new dynamically.*
DELETE	• Removal of an object record, where supported in the application; Or **Delete**.

The default method is GET but an API user has a choice of any of the four methods to access and control resources that is supported by Taleo Business Edition.

URL STRUCTURE

Understanding the Taleo REST API URL structure is essential to integration success. The REST API is made up of a host URL and can have resource parameters defined in at least two ways:

- As part of the URL-path (i.e. /resource/parametervalue)
- As a query argument (i.e. /resource?parameter=value)

Every request must be sent to a single specific Taleo Business Edition instance via a host URL. To find this URL, you must engage the Taleo Dispatcher Service. The Dispatcher provides one simple function: request the Taleo unique instance ID (company code) and receive the host URL to send resource requests to.

https://tbe.taleo.net/MANAGER/dispatcher/api/v1/serviceUrl/**<<COMPANY_CODE>>{.xml}**

Once the host URL has been determined for a specific Taleo instance, you must then login for service and receipt of session authToken. The REST API requires submission of username, password and company code credentials as part of login resource request as request body:

<<HOST_URL>>/login

request body

orgCode=<<COMPANY_CODE>>&userName=<<USER_NAME>>&password=<<PASSWORD>>

The returning session authToken value provided has to be set in subsequent calls as part of the header (as a cookie). The rest of the URL structure will vary based on the core action you would like to take with the resource.

The actions supported by the Taleo Business Edition REST API include:

- Get
- Get All
- Search
- Create / Upsert
- Update
- Delete

The below table provides a general structure of URL format based on action and method required.

Method	Action	URL Format & Example	System Parameters
GET	**Get** single resource	…/object/**<<ENTITY_NAME>>**/**<<ENTITY_ID>>**{.xml} eg (…/object/division/1)	
GET	**Get All** (where search not supported)	…/object/**<<ENTITY_NAME>>**{.xml} eg (../object/division)	
GET	**Search** for resource(s)	…/object/**<<ENTITY_NAME>>**/search{.xml} eg(…/object/account/search?parameter=xxx&…)	• **?addedWithin** XX number of days for overall record creation date searching • **?updatedWithin** XX number of days for overall record creation date searching • **?<fieldname>_from** and **<fieldname>_to** for searching between integer fields • **?<fieldname>_from** and **<fieldname>_to** for searching between date fields • **?<fieldname>_from** and **<fieldname>_to** for searching between currency fields • **?fields=**xxx**|**xxx**|** for filtering the results by specific field(s) separated by pipe "|" • **?view=links** for filtering the results by displaying related links only • **?start=1&limit=5** for pagination of results in search response
PUT	**Update** resource	…/object/**<<ENTITY_NAME>>**/**<<ENTITY_ID>>**{.xml} eg (…/object/companygoal/123)	
DELETE	**Delete** resource	…/object/**<<ENTITY_NAME>>**/**<<ENTITY_ID>>**{.xml} eg (…/object/department/123)	
POST	**Create** resource	…/object/**<<ENTITY_NAME>>**{.xml} eg (…/object/account)	
POST	**Upsert**: Create resource unless it exists then update.	…/object/**<<ENTITY_NAME>>**{.xml}?upsert=true eg (…/object/account?upsert=true)	• **?upsert=true** which flags validation for updating if existing record already exists.

AUTHENTICATION AND AUTHORIZATION

Taleo Business Edition REST APIs support the following authentication method:

Cookie base authentication:

This method requires submission of username, password and company code credentials as part of login resource request. The returning session value provided has to be set in subsequent calls as part of the cookie within the header. Configured as: authToken=<<webapi_token>>.

REQUEST FORMAT & HEADER

The Taleo Business Edition REST API uses HTTP as the request protocol for engaging with a single Taleo Business Edition instance. The request message format would include:

- A Request-Line:
  - Method e.g. GET;
  - URL (JSON by default but .xml attribute for XML) e.g …/object/resource …/object/resource**.xml**
- A Header:
  - authToken e.g. Cookie authToken=webapixxxxxxxxxxxxxxxxx
  - Content-type of the representation, or format of the data (depending on the request) e.g. Content-Type application/json or application/xml
- A Message Body depending on the request method

As long as the session (Cookie) is still valid, all requests should include all of the necessary details to process the request which makes REST API's so powerful. The stateless communication makes REST API much more engaging then SOAP.

REQUEST MESSAGE BODY

The POST or PUT methods in the Taleo Business Edition REST API allow a user to submit a representation to the provided URL. This would require inclusion of a message body. The message body would be used for creating (POSTing) or updating (PUTing) of an object record dependent on the method submitted.

The data provided in the message body would need to meet the minimum requirements of that specific resource for successful submission. For example, when creating a new employee record Taleo Business Edition requires an employeeNumber and lastName attribute for successful record creation. If that minimum requirement has not been submitted in the message body of a POST request, an appropriate error notification would be in response to notify user of issue.

To minimize the amount of error notifications, when non-editable fields are passed through the message body, Taleo Business Edition API does disregard those elements as best as possible. Non-editable fields are typically derived system fields like: ID, lastUpdatedDate, and creationDate.

Reminder: POST is the creation of a resource in Taleo Business Edition, while PUT is an update of that resource. Taleo Business Edition includes required fields/format/varibles on PUT and POST that may need to be included in your message body for successful request transmission. Refer to your Taleo Business Edition instance for details or in the sample code section of this document.

Message body's in JSON start with the entity tag and then the field data, while in XML they are surrounded by <root> tags and then the entity tag. This is consistent throughout the API:

JSON

httpHeader: Content-Type application/json Cookie authToken=webapi2-3759841272XXXXXX
url: **<<HOST_URL>>**/object/account
method: POST

```
{
  "account":{
    "name":"webapibwbjbmubfrkshdqjqfqd",
    "city":"webapingetwzzrtzbmframxwpd",
    "address":"webapitbcpxtdgtnxcpjkpszdh",
    "websiteURL":"webapihqnuawufdspqsxeuupyv",
    "phone":"4879353283",
    "fax":"6538339942",
    "state":"US-CA",
    "industry":"Automotive"
  }
}
```

XML

httpHeader: Content-Type application/xml Cookie authToken=webapi2-3759841272XXXXXX
url: **<<HOST_URL>>**/object/account.xml
method: POST

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
 <account>
  <websiteURL>webapirdvjhjnymfpvhdhdpbkn</websiteURL>
  <address>webapijxyduvycpfawretvvumx</address>
  <city>webapiybtcpncvehjqpjsuzvgv</city>
  <name>webapiukgcmdfpusdxhsgkhbau</name>
  <fax>6728156136</fax>
  <phone>3716413771</phone>
  <industry>Automotive</industry>
  <state>US-CA</state>
 </account>
</root>
```

RESPONSE NOTIFICATIONS & ERROR CODES

If a request to the Taleo Business Edition REST API is successful, it returns the requested data in structured JSON or XML with a response that indicates success:

JSON

```
"status":{
    "detail":{

    },
    "success":true
  }
```

XML

```
<root>
 <status>
  <detail/>
  <success>true</success>
 </status>
</root>
```

If a request fails, no data is returned, and the service returns an HTTP status code. If the failure occurred with Taleo Business Edition, the status code is returned with a diagnostic message to assist in fixing the issue, for example:

JSON

```
{
  "response":{
  },
  "status":{
    "detail":{
      "operation":"internal",
      "errormessage":"Content that does not conform to JSON/XML syntax as per specification",
      "error":"Could not read [class com.rf.common.integration.rest.RestDisplayFieldValueWrapper]",
      "errorcode":"500"
    },
    "success":false
  }
}
```

XML

```
<root>
 <response/>
 <status>
  <detail>
   <operation>internal</operation>
   <errormessage>Content that does not conform to JSON/XML syntax as per specification</errormessage>
   <error>Could not read [class com.rf.common.integration.rest.RestDisplayFieldValueWrapper]</error>
   <errorcode>500</errorcode>
  </detail>
  <success>false</success>
 </status>
</root>
```

Where the following issues may have occurred:

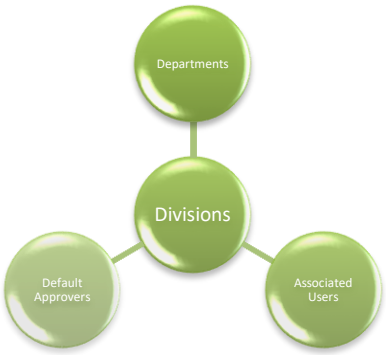
Error Code	Message	Detailed Sub Message
500	Rest API call failed with exception	
400	Bad request	
401	Service not enabled for customer	• "User **<<username>>** does not have access to this API. Only admin users are allowed to access the APIs"
403	Forbidden Request	• "Customer is not a premium level customer to use API" • "Service not enabled for the customer"
404	Requested resource not found	
412	Validation failure on request	• REQUIRED_FIELD_MISSING: ErrorMessage="Required field is missing" • SUPPLIED_FIELD_DOES_NOT_EXIST_ON_OBJECT: ErrorMessage="Supplied field does not exist on object "

		• WRONG_DATA_TYPE: ErrorMessage=”Wrong data type” • INVALID_PICKLIST_VALUE: ErrorMessage=”Picklist value provided is invalid” • INVALID_REVIEW_TEMPLATE: ErrorMessage=”Invalid performance review template” • INVALID_MULTI_PICKLIST_VALUE: ErrorMessage=”One or more picklist value provided is invalid” • INVALID_LOCALE_VALUE: ErrorMessage=”Invalid locale value” • INVALID_USER_ROLE_VALUE: ErrorMessage=”User role provide is invalid” • INVALID_USER_EMAIL: ErrorMessage=”Invalid email or given email is already used by other user” • INVALID_EMPLOYEE_EMAIL: ErrorMessage=”Invalid email or given email is already used by other user” • INVALID_USER_LOGIN: ErrorMessage=”Invalid user login id or given login id is already used by other user” • INVALID_EMPLOYEE_LOGIN: ErrorMessage=”Invalid employee login id or given login id is already used by other employee” • EXCEEDED_MAX_ACTIVE_USER_LICENCE: ErrorMessage=”Exceeded maximum active user licence” • EXCEEDED_MAX_PASSIVE_USER_LICENCE: ErrorMessage=”Exceeded maximum passive user licence” • INVALID_TIMEZONE_VALUE: ErrorMessage=”Invalid timezone value” • TEXT_VALUE_TOO_LONG: ErrorMessage=”Provided value for text is too long” • URL_VALUE_TOO_LONG: ErrorMessage=”Provided value for url is too long” • INVALID_DISPLAY_FIELD_TYPE: ErrorMessage=”invalid display field type” • INVALID_ENTITY_TYPE: ErrorMessage=”invalid entity type” • INVALID_DISPLAY_FIELD_NAME: ErrorMessage=”invalid display field name” • INVALID_EXTERNAL_NAME: ErrorMessage=”invalid external name” • INVALID_REGION_NAME: ErrorMessage=”Region with given name does not exist” • INVALID_DEPARTMENT_NAME: ErrorMessage=”Department with given name does not exist” • INVALID_DEPARTMENT_ID: ErrorMessage=”Invalid department id” • SPACE_IN_EXTERNAL_NAME: ErrorMessage=”Blank space character not allowed in external name” • DISPLAY_FIELD_ALREADY_EXIST: ErrorMessage=”Display field with same name for the given entity already exists” • REGION_ALREADY_EXIST: ErrorMessage=”Region with given name already exist. Duplicate region not allowed” • DEPARTMENT_ALREADY_EXIST: ErrorMessage=”Department with given name already exist. Duplicate department not allowed” • DISPLAY_FIELD_NOT_EXIST: ErrorMessage=”Display field with given name for the given entity does not exists” • DISPLAY_FIELD_NOT_ALLOWED_FOR_UPDATE: ErrorMessage=”Display field with given name for the given entity is not allowed for update. Display field of type Picklist are only allowed for update” • INVALID_ASSOCIATED_LOCATION: ErrorMessage=”One or more location provided is invalid” • INVALID_ASSOCIATED_USER: ErrorMessage=”One or more associated users provided is invalid” • INVALID_DEFAULT_APPROVER: ErrorMessage=”One or more approvers provided is invalid” • INVALID_ACCOUNT_OWNER: ErrorMessage=”One or more owners provided is invalid” • INVALID_MANAGER_ID: ErrorMessage=”Invalid manager id” • INVALID_MANAGER_ID: ErrorMessage=”Invalid manager id” • INVALID_WORKFLOW: ErrorMessage=”Invalid workflow definition” • INVALID_USER_ID: ErrorMessage=”Invalid user id” • INVALID_PAY_RANGE: ErrorMessage=”Invalid pay range” • INVALID_EMPLOYEE_ID: ErrorMessage=”Invalid employee id” • EMPLOYEE_ID_ALREADY_ASSOCIATED: ErrorMessage=”Employee id provided is already associated with another user”

		• INVALID_EMPLOYEE_CODE: ErrorMessage="Only alpha numeric and dashes are allowed in Employee code" • DUPLICATE_EMPLOYEE_CODE: ErrorMessage="Employee code provided is already associated with another employee" • INVALID_STATUS: ErrorMessage="Invalid status" • FIELD_IS_NOT_SEARCHABLE: ErrorMessage="Field is not searchable" • INVALID_REGION_ID: ErrorMessage="Invalid regionId" • INVALID_DIVISION_ID: ErrorMessage="Invalid division id" • INVALID_LOOKUP_VALUE: ErrorMessage="Invalid lookup value" • LOCATION_ALREADY_EXIST: ErrorMessage="Location with given name already exist. Duplicate location not allowed" • INVALID_LOCATION_ID: ErrorMessage="Invalid location id" • INVALID_LOCATION_NAME: ErrorMessage="Location with given name does not exist" • DIVISION_ALREADY_EXIST: ErrorMessage="Division with given name already exist. Duplicate division not allowed" • INVALID_DIVISION_NAME: ErrorMessage="Division with given name does not exist" • INVALID_ASSOCIATED_DEPARTMENT: ErrorMessage="One or more departments provided is invalid" • INVALID_COMPANY_GOAL: ErrorMessage="Invalid company goal id" • NON_ACTIVE_COMPANY_GOAL: ErrorMessage="Company goal is not an active goal" • NON_ACTIVE_EMPLOYEE_GOAL: ErrorMessage="Employee goal is not an active goal" • INVALID_EMPLOYEE_GOAL: ErrorMessage="Invalid employee goal id" • INVALID_SCORE_VALUE: ErrorMessage="Invalid or incorrect score" • UNAVAILABLE_ASSOCIATED_DEPARTMENT: ErrorMessage="One or more departments provided is unavailable"

RESPONSE RELATIONSHIP URLS

Some resources provide specialized URLs for related entities. This quick action function allows the API user to request an additional task or item to that resource that has an inherent relationship.



For example, a division assignment in Taleo Business Edition is to departments, users and default approvers for requisitions. There are three related entities to a division resource. The Taleo Business Edition API provides the core response data as part of the GET request but also includes direct relational URL's to receive related URL's.

The standard URL format for GET is:

<<HOST_URL>>/object/**<<OBJECT_ENTITY_NAME>>**

However when specific for a relationship to another entity, the format becomes:

<<HOST_URL>>/object/**<<OBJECT_ENTITY_NAME>>**/**<<OBJECT_ENTITY_ID>>**/**<<RELATIONSHIP_OBJECT_ENTITY>>**

The result of that direct URL is the specific results associated with the primary object GET request.

For example, when querying for divisions which have 3 departments assigned to them, my direct relationshipUrl will have the full record output of the 3 related departments.

Sample code provided below for reference:

JSON


```
{
  “response”:{
    “divisions”:[
      {
        “id”:1,
        “divisionName”:”Division1”,
        “creationDate”:”2011-10-05”,
        “lastUpdated”:”2011-11-18”,
        “requisitionApprovers”:[
          15,
          39
        ],
        “description”:””,
        “entityType”:”DIVSN”,
        “associatedUsers”:[
          15,
          40
        ],
        “associatedDepartments”:[
          2
        ],
        “relationshipUrls”:{
          “associatedDepartments”:”https://tbe.taleo.net/NA2/ats/api/v1/object/division/1/associatedDepartments”,
          “requisitionApprovers”:”https://tbe.taleo.net/NA2/ats/api/v1/object/division/1/requisitionApprovers”,
          “associatedUsers”:”https://tbe.taleo.net/NA2/ats/api/v1/object/division/1/associatedUsers”
        }
      }
    ]
  },
  “status”:{
    “detail”:{

    },
    “success”:true
  }
}
```

XML

```
<root>
 <response>
  <divisions>
   <item>
    <id>1</id>
    <divisionName>Division1</divisionName>
    <creationDate>2011-10-05</creationDate>
    <lastUpdated>2011-11-18</lastUpdated>
    <requisitionApprovers>
     <item>15</item>
     <item>39</item>
    </requisitionApprovers>
    <description></description>
    <entityType>DIVSN</entityType>
    <associatedDepartments>
     <item>2</item>
    </associatedDepartments>
    <associatedUsers>
     <item>15</item>
     <item>40</item>
    </associatedUsers>
    <relationshipUrls>
<associatedDepartments>https://tbe.taleo.net/NA2/ats/api/v1/object/division/1/associatedDepartments.xml</associatedDepartments>
     <requisitionApprovers>https://tbe.taleo.net/NA2/ats/api/v1/object/division/1/requisitionApprovers.xml</requisitionApprovers>
     <associatedUsers>https://tbe.taleo.net/NA2/ats/api/v1/object/division/1/associatedUsers.xml</associatedUsers>
    </relationshipUrls>
   </item>
  </divisions>
 </response>
 <status>
  <detail/>
  <success>true</success>
 </status>
</root>
```

ENTITY OVERVIEW

The Taleo Business Edition platform is made up of multiple licensed modules that can be purchased by customers at any time. These modules include Taleo Business Edition:

- Recruit
- OnBoard
- Perform, and
- Comp

As a single platform, as additional modules are purchased or 'turned on', the REST API will automatically enable programmatic access to the additional API's. Caveat being, you may need to logout of your current session and login again for a refresh.

Under each module there is a primary system entity or table(s) of records and then sub system entities. For example, Candidate table is a primary system entity, however a sub table that is related to candidates can include work history, education, background checks, etc. All primary system entities include a search function within the application which has been replicated within the API. The search function is used to query for relevant data that is required. Sub-entities do not include a search function, but in some cases may include a 'get-all' function for ease of receiving a batch output.

Refer below for a listing of search included entities and a listing of all entities available. Included is the module that needs to be purchased in order to access the appropriate entity / sub-entity.

Keep reference of the Entity Code value in the entity tables below as these can be used for grabbing metadata and status data through the API via additional REST interactions.

Primary System Entities:

Entity Name	Entity Code	Entity Function	TBE Product / Module	URL to get Collection
account	ACCT	Used for managing external recruiting accounts	Recruit	object/account/search
candidate	CAND	Used for storing and managing candidates	Recruit	object/candidate/search
employee	EMPL	Used for storing and managing employees	Onboard/Perform/Comp	object/employee/search
requisition	REQU	Used for managing requisitions and requisition templates	Recruit	object/requisition/search
user	WORK	Workforce users of the system (platform users)	ANY/ALL MODULES	object/user/search

All API Supported System Entities:

Entity Name	Entity Code	Entity Function	TBE Product / Module
account	ACCT	Used for managing external recruiting accounts	Recruit
activity	ACTV	Activities for employees to complete as part of onboarding	Onboard
Attachment (see Appendix A for information about attachments)	ATCHM	Used for storing record level attachments	ANY/ALL MODULES
Answer	ANSWE	Used for storing responses to screening questions	Recruit
Approval	APPR	Used for storing approval or rejection of an offer or requisition	Recruit
backgroundcheck	BKGRC	Used for storing background checks at the candidate level	Recruit
candidate	CAND	Used for storing and managing candidates	Recruit
candidateapplication	CAREQ	Relationship Table used for mapping candidates to requisitions	Recruit

careerwebsite		Used by poster entity for manage posting status	Recruit
certificate	CERTIFICATES_LICENCES	Candidate **OR** Employee running log of certificate & License history (Application collection)	ANY/ALL MODULES
comment	COMT	Cross system Comment Logging	ANY/ALL MODULES
companygoal	CGOAL	Used for enterprise wide corporate goals	Perform
competency	COMP	Library used for competency storage	Perform
contact	CTCT	Used for managing external recruiting contacts	Recruit
Contactlog	CONLG	Used for a running trail for communcations	ANY/ALL MODULES
department	DEPT	Used for storing all enterprise departments	ANY/ALL MODULES
displayfield	DISPLAYFIELD	Each field in the system	ANY/ALL MODULES
division	DIVSN	Used for storing all enterprise divisions (master grouping of departments)	ANY/ALL MODULES
education	EDUCATION	Candidate **OR** Employee running log of education history (Application collection)	ANY/ALL MODULES
employee	EMPL	Used for storing and managing employees	Onboard/Perform
employeegoal	GOAL	Employee goal store	Perform
entitylink	ENTITYLINK	Linkage between one entity record to another entity record for display & reporting	ANY/ALL MODULES
expense	EXPS	Expenses that pertain to a source, requisition or candidate	Recruit
Historylog	HIST	Used to store running history of updates to other entities	ANY/ALL MODULES
interview	INTV	Candidate Interview Log	Recruit
interviewfeedback	FEED	Candidate interview feedback	Recruit
location	LOCA	Used for storing all enterprise locations	ANY/ALL MODULES
offer	OFFER	Offers to candidates against requisitions	Recruit
packet	ACTP	Logical grouping or bundle of activities for onboarding	Onboard
parseresume		Used for sending resume files to 3rd party resume parsing engine	Recruit
pmreview	PMRV	Performance Review store for employees	Perform
poster	POSTER	Requisition posting to Career Websites	Recruit
question	QUEST	Question library of screening questions that are attached to requisitions	Recruit
reference	REFERENCES	Candidate **OR** Employee running log of references history (Application collection)	ANY/ALL MODULES
region	REGIO	Used for storing all enterprise regions (master grouping of locations)	ANY/ALL MODULES
requisition	REQU	Used for managing requisitions and requisition templates	Recruit
residence	RESIDENCE_HISTORY	Candidate **OR** Employee running log of residence history (Application collection)	ANY/ALL MODULES
resumetocandidate		Used for sending resume to 3rd party parsing engine and creating candidate record	Recruit
status	STATS	For record level status definition (i.e. pending, active, inactive)	ANY/ALL MODULES
user	WORK	Workforce users of the system (platform users)	ANY/ALL MODULES
workhistory	WORK_HISTORY	Candidate **OR** Employee running log of work history (Application collection)	ANY/ALL MODULES
candidateanswer	CandTest	Used for storing and managing candidate answers to screening questions for a requisition	Recruit

GETTING STARTED

Getting started with the Taleo Business Edition REST API for the first time requires a discovery process that includes:

Finding where your Taleo Business Edition instance (Host / EndPoint) is located in our server pools:

Location	End Point Examples
United States	https://chm.tbe.taleo.net/chmXX/ats/api/
Europe	https://ldd.tbe.taleo.net/lddXXX/ats/api/
Canada	https://trr.tbe.taleo.net/trrXXX/ats/api/

Followed by logging into your Taleo instance, requesting to see what services are available via the REST API and then lastly grabbing the META data associated with the object you are interested in.

Please Note: Taleo reserves the right to move on customer to another server or data center at any given time for improved data security, load balancing, maintenance, etc. As such, we recommend prior to every usage of the API, you query for your host instance.

Finding Your Instance:

The Dispatcher provides one simple function: request the organization code and receive a response URL of where the account/zone is located in the pool of Taleo Business Edition servers. The reason for the Dispatcher is that accounts/zones may be moved to a new server pool for optimal load balancing. The Dispatcher ensures that the client application will always find the organization URL.

JSON

Method:	Request:
GET	https://tbe.taleo.net/MANAGER/dispatcher/api/v1/**serviceUrl/<<COMPANY_CODE>>**

	Response:
	
	{
	“response”:{
	“URL”:”https://**<<DATA_CENTER_LOCATION>>/<<SERVER_NAME>>**/ats/api/v1/”
	},
	“status”:{
	“detail”:{
	
	},
	“success”:true
	}
	}

XML

Method:	Request:
GET	https://tbe.taleo.net/MANAGER/dispatcher/api/v1/**serviceUrl/<<COMPANY_CODE>>**.xml
	Response: <root> <response> <URL>https://**<<DATA_CENTER_LOCATION>>/<<SERVER_NAME>>**/ats/api/v1.xml</URL> </response> <status> <detail/> <success>true</success> </status> </root>

Requesting Login/Logout:

Cookie Based Authentication

When using the cookie based authentication method, the Taleo Business Edition REST API uses cookie- based authentication through a saved authToken variable. This requires you to first submit (i.e. POST) your login credentials to the API, grab your session key and then save in the cookie.

The login process replicates the frontend application with three parameters required: username, password and company code (unique Taleo customer value). If you do not have these variables, please contact your Taleo Administrator to receive these. You will also need the permission role of Administrator to access the REST API entirely. Your user role can also have encrypted value access through the API (i.e. unmasked values of SSN, DOB and custom encrypted fields) if your user role has 'Access to Confidential Data' marked true.

Each login credential has a maximum of 20 active connections, and each new login is issued a new authToken. AuthTokens expire in 4 hours.

Therefore, 20 consecutive login calls to the API could effectively consume all 20 available tokens and hold them open for a 4 hour period. When all 20 tokens are active, no new API login are possible until the active authTokens are logged out or expire. **It is highly recommended that all transactions to the API begin with a login and end with a logout**.

Parameters:

Parameter	Value Type	Description
orgCode	String	Unique Taleo Business Edition company instance, referred to as a company code.
Username	String	Unique username of a Taleo Business Edition user. Please Note, this user must be an Administrator.
Password	String	User's unique password.

REST API is available for Taleo Business Edition Premium Service level customers. Users defined as System Administrator roles can gain access for login, and only defined 'Security Managers' will be able to retrieve encrypted data unmasked (Social Security Numbers, Date of Birth, etc.)

To login:

JSON

Method:	Request:
POST	**<<HOST_URL>>/login** **request body :** orgCode=**<<COMPANY_CODE>>**&userName=**<<USER_NAME>>**&password=**<<PASSWORD>>**

	Response:
	
	{
	“response”:{
	“authToken”:”**<<AUTH_TOKEN>>**”
	},
	“status”:{
	“detail”:{
	
	},
	“success”:true
	}
	}

XML

Method:	Request:
POST	**<<HOST_URL>>/login** **request body :** orgCode=**<<COMPANY_CODE>>**&userName=**<<USER_NAME>>**&password=**<<PASSWORD>>**
	Response: <root> <response> <authToken>**<<AUTH_TOKEN>>**</authToken> </response> <status> <detail/> <success>true</success> </status> </root>

To logout:

JSON

Method:	Request:
POST	**<<HOST_URL>>/logout**
	Response: {

<table>
<tr><td></td><td>"response":
{
},
"status":
{
"detail":
{
},
"success": true
}
}</td></tr>
</table>

XML

<table>
<tr><td>Method:

POST</td><td>Request:

<<HOST_URL>>/logout.xml</td></tr>
<tr><td></td><td>Response:

<root>
<response/>
<status>
<detail/>
<success>true</success>
</status>
</root></td></tr>
</table>

Discovery of Services:

The Get Objects request provides the ability to programmatically receive all the API entities available through the Taleo API. For the TBE Classic REST API's, you will receive an array of all objects plus the object definitions and urls for building out your API calls. For the TBE JSON Hyperschema API's, you will receive a list of hyperschemas with a reference to the hyperschema description. This is the most important call of the Taleo Business Edition RESTful service.

Parameters:

Parameter	Value Type	Description
name	String	Object Entity Name (API representation)
label	String	Object Entity Label Name (GUI / UI representation)
searchable	Boolean	Whether the Object supports searching (includes a search API) or whether only a get all API call is supported.
Creatable	Boolean	Whether the Object supports creating records (includes a PUT/POST API) or whether only a GET API call is supported.
Deletable	Boolean	Whether the Object supports deleting records (includes a DELETE API).
Urls	Array	An array of Object specific URLs to get to the next level: • GET single object info • GET meta data info for standard fields • GET meta data info for custom fields • GET a single record URL • GET a collection of records URL

You can choose to receive a single object versus all by using the object/info/ URL but appending the object entity name as the last URL item (**<<HOST_URL>>**/object/info**<<OBJECT_ENTITY_NAME>>**)

TBE Classic REST API JSON

Method:	Request:
POST	**<<HOST_URL>>**/object/info

Response:


```
{
  “response”:{
    “objects”:[
      {
        “name”:”<<OBJECT_ENTITY_NAME>>”,
        “label”:”<<OBJECT_LABEL_NAME>>”,
        “searchable”:”<<OBJECT_ SEARCHABLE_BOOLEAN-VALUE>>”,
        “creatable”:”<<OBJECT_CREATABLE_BOOLEAN-VALUE>>”,
        “deletable”:”<<OBJECT_DELETABLE_BOOLEAN-VALUE>>”,
        “urls”:{
          “object”:”<<HOST_URL>>/object/info<<OBJECT_ENTITY_NAME>>”,
          “standard-description”:”<<HOST_URL>>/object/<<OBJECT_ENTITY_NAME>>/description/standard”,
          “custom-description”:”<<HOST_URL>>/object/<<OBJECT_ENTITY_NAME>>/description/custom”,
          “instance”:”<<HOST_URL>>/object/<<OBJECT_ENTITY_NAME>>/{id}”,
          “collection”:”<<HOST_URL>>/object/<<OBJECT_ENTITY_NAME>>/{search}”
        }
      },
       {
        “name”:”<<OBJECT_ENTITY_NAME>>”,
        “label”:”<<OBJECT_LABEL_NAME>>”,
        “searchable”:”<<OBJECT_ SEARCHABLE_BOOLEAN-VALUE>>”,
        “creatable”:”<<OBJECT_CREATABLE_BOOLEAN-VALUE>>”,
        “deletable”:”<<OBJECT_DELETABLE_BOOLEAN-VALUE>>”,
        “urls”:{
          “object”:”<<HOST_URL>>/object/info<<OBJECT_ENTITY_NAME>>”,
          “standard-description”:”<<HOST_URL>>/object/<<OBJECT_ENTITY_NAME>>/description/standard”,
          “custom-description”:”<<HOST_URL>>/object/<<OBJECT_ENTITY_NAME>>/description/custom”,
          “instance”:”<<HOST_URL>>/object/<<OBJECT_ENTITY_NAME>>/{id}”,
          “collection”:”<<HOST_URL>>/object/<<OBJECT_ENTITY_NAME>>/{search}”
        }
      },
```


ETC. ETC.


```
        }
      }
    ]
  },
  “status”:{
    “detail”:{

    },
    “success”:true
  }
}
```


TBE Classic REST API XML

Method:	Request:
POST	**<<HOST_URL>>**/object/info.xml

Response:

```
<root>
 <response>
  <objects>
   <item>
    <name><<OBJECT_ENTITY_NAME>></name>
    <label><<OBJECT_LABEL_NAME>></label>
    <searchable><<OBJECT_ SEARCHABLE_BOOLEAN-VALUE>></searchable>
    <creatable><<OBJECT_CREATABLE_BOOLEAN-VALUE>></creatable>
    <deletable><<OBJECT_DELETABLE_BOOLEAN-VALUE>></deletable>
    <urls>
     <object><<HOST_URL>>/object/info<<OBJECT_ENTITY_NAME>>.xml</object>
     <standard-description><<HOST_URL>>/object/<<OBJECT_ENTITY_NAME>>/description/standard.xml</standard-
description>
     <custom-description><<HOST_URL>>/object/<<OBJECT_ENTITY_NAME>>/description/custom.xml</custom-description>
     <instance><<HOST_URL>>/object/<<OBJECT_ENTITY_NAME>>/{id}.xml</instance>
     <collection><<HOST_URL>>/object/<<OBJECT_ENTITY_NAME>>/{search}</collection>
    </urls>
   </item>
   <item>
    <name><<OBJECT_ENTITY_NAME>></name>
    <label><<OBJECT_LABEL_NAME>></label>
    <searchable><<OBJECT_ SEARCHABLE_BOOLEAN-VALUE>></searchable>
    <creatable><<OBJECT_CREATABLE_BOOLEAN-VALUE>></creatable>
    <deletable><<OBJECT_DELETABLE_BOOLEAN-VALUE>></deletable>
    <urls>
     <object><<HOST_URL>>/object/info<<OBJECT_ENTITY_NAME>>.xml</object>
     <standard-description><<HOST_URL>>/object/<<OBJECT_ENTITY_NAME>>/description/standard.xml</standard-
description>
     <custom-description><<HOST_URL>>/object/<<OBJECT_ENTITY_NAME>>/description/custom.xml</custom-description>
     <instance><<HOST_URL>>/object/<<OBJECT_ENTITY_NAME>>/{id}.xml</instance>
     <collection><<HOST_URL>>/object/<<OBJECT_ENTITY_NAME>>/{search}</collection>
    </urls>
   </item>
```

ETC.ETC.

```
  </objects>
 </response>
 <status>
  <detail/>
  <success>true</success>
 </status>
</root>
```

Method:	Request:
POST	**<<HOST_URL>>**/object/info/hyperschema

Response:


```
{
 "items": [
  {
   "name": "Account",
   "links": [
    {
     "href": "https://dcb.tbe.taleo.net/DCB02/ats/api/v1/object/account/hyperschema.json",
     "rel": "describes",
     "mediaType": "application/schema+json"
    }
   ]
  },
  {
   "name": "Candidate",
   "links": [
    {
     "href": "https://dcb.tbe.taleo.net/DCB02/ats/api/v1/object/candidate/hyperschema.json",
     "rel": "describes",
     "mediaType": "application/schema+json"
    }
   ]
  },
  {
   "name": "WorkhistoryCandidate",
   "links": [
    {
     "href": "https://dcb.tbe.taleo.net/DCB02/ats/api/v1/object/candidate/workhistory/hyperschema.json",
     "rel": "describes",
     "mediaType": "application/schema+json"
    }
   ]
  },
  {
   "name": "EducationCandidate",
   "links": [
    {
     "href": "https://dcb.tbe.taleo.net/DCB02/ats/api/v1/object/candidate/education/hyperschema.json",
     "rel": "describes",
     "mediaType": "application/schema+json"
    }
   ]
  },
  {
   "name": "CertificateCandidate",
   "links": [
    {
     "href": "https://dcb.tbe.taleo.net/DCB02/ats/api/v1/object/candidate/certificate/hyperschema.json",
     "rel": "describes",
     "mediaType": "application/schema+json"
    }
   ]
  },
  {
   "name": "ResidenceCandidate",
   "links": [
    {
     "href": "https://dcb.tbe.taleo.net/DCB02/ats/api/v1/object/candidate/residence/hyperschema.json",
     "rel": "describes",
     "mediaType": "application/schema+json"
```

```
    }
   ]
  },
  {
   "name": "ReferenceCandidate",
   "links": [
    {
     "href": "https://dcb.tbe.taleo.net/DCB02/ats/api/v1/object/candidate/reference/hyperschema.json",
     "rel": "describes",
     "mediaType": "application/schema+json"
    }
   ]
  },
  {
```


ETC.ETC.

Using Search

Default limit

When a search is initiated using the APIs, a limit of 200 is applied by default if one is not explicitly supplied in the URL.

../api/v1/object/employee/search?limit=200

If the total result set is more than 200 records (or the limit supplied), a relationship URL is returned in the response, in the 'next' tag, that can be used to retrieve the next set of records for the search.

Searching by Timestamps

When performing a search using a parameter that is a timestamp, for example lastUpdated, the values supplied to the search parameters must follow the format of the full timestamp.

Example:

- …/search?lastUpdated_from=2013-01-09T13:49PST&lastUpdated_to=2013-09-09T13:49PST
- …/search?lastUpdated_from=2013-01-09T13:49PST

Searching by Text fields

When performing a search using a parameter that is a Text data type, the search performed is a **LIKE** for the value supplied. The search does not perform an exact match.

The result returned will be for any object where the value provided for the parameter appears anywhere in the field specified.

Care should be taken to inspect the result set returned for the exact record if an exact match is required.

Discovery of Metadata

The Get Metadata request provides ability to programmatically receive all the data associated with a specific object. For ease of reference, you are able to pull all standard system fields only (those shared by all Taleo customers) versus including custom configured fields as well. Note this option is only available for

the TBE Classic REST API's; the JSON Hyperschema Get Metadata request includes both standard and custom fields.

All Taleo Business Edition customers have the ability to rename standard fields, however customers can create, update and delete custom fields as they wish through the Taleo Business Edition Administration panel.

When retrieving Metadata for TBE Classic REST API's, please note you are able to pull standard fields only by using a URL parameter labeled 'standard'. To pull standard and custom fields, use the 'custom' parameter. The difference between both URLs will be: <<HOST_URL>>/object/<<OBJECT_LABEL_NAME>>/description/**standard** vs <<HOST_URL>>/object/<<OBJECT_LABEL_NAME>>/description/**custom**.

TBE Classic REST API JSON

Method:	Request:
GET	**<<HOST_URL>>**/object/**<<OBJECT_LABEL_NAME>>**/description/**{"standard" or "custom"}**

Response:

```
{
  "response":{
    "code":"<<OBJECT_ENTITY_CODE>>",
    "name":"<<OBJECT_ENTITY_NAME>>",
    "label":"<<OBJECT_LABEL_NAME>>",
    "fields":[
      {
        "name":"<<OBJECT_FIELD_EXTERNAL-API_NAME>>",
        "label":"<<OBJECT_FIELD_GUI-LABEL_FIELD_NAME>>",
        "isCustom":><<OBJECT_FIELD_CUSTOM-OR-STANDARD_BOOLEAN-VALUE>>",
        "30vailabl":"<<OBJECT_FIELD_FIELD-TYPE>>",
        "isSearchable":<<OBJECT_FIELD_SEARCHABLE_BOOLEAN-VALUE>>,
        "isRequired":<<OBJECT_FIELD_REQUIRED-OR-NOT_BOOLEAN-VALUE>>,
        "isReadOnly":<<OBJECT_FIELD_EDITABLE-OR-NOT_BOOLEAN-VALUE>>,
        "description":"<<OBJECT_FIELD_DESCRIPTION-VALUE>>"
      },
      {
        "name":"<<OBJECT_FIELD_EXTERNAL_NAME>>",
        "label":"<<OBJECT_FIELD_INTERNAL-GUI_NAME>>",
        "isCustom":><<OBJECT_FIELD_CUSTOM-OR-STANDARD_BOOLEAN-VALUE>>",
        "30vailabl":"<<OBJECT_FIELD_FIELD-TYPE>>",
        "isSearchable":<<OBJECT_FIELD_SEARCHABLE_BOOLEAN-VALUE>>,
        "isRequired":<<OBJECT_FIELD_REQUIRED-OR-NOT_BOOLEAN-VALUE>>,
        "isReadOnly":<<OBJECT_FIELD_EDITABLE-OR-NOT_BOOLEAN-VALUE>>,
        "description":"<<OBJECT_FIELD_DESCRIPTION-VALUE>>"
      }
    ]
  },
  "status":{
    "detail":{

    },
    "success":true
  }
}
```

TBE Classic REST API XML

Method:	Request:
GET	**<<HOST_URL>>**/object/**<<OBJECT_LABEL_NAME>>**/description/**{"standard" or "custom"}**.xml
	Response:

```
<root>
 <response>
  <code><<OBJECT_ENTITY_CODE>></code>
  <name><<OBJECT_ENTITY_NAME>></name>
  <label><<OBJECT_LABEL_NAME>></label>
  <fields>
   <item>
    <name><<OBJECT_FIELD_EXTERNAL-API_NAME>></name>
    <label><<OBJECT_FIELD_INTERNAL-GUI_NAME>></label>
    <isCustom><<OBJECT_FIELD_CUSTOM-OR-STANDARD_BOOLEAN-VALUE>></isCustom>
    <31vailabl><<OBJECT_FIELD_FIELD-TYPE>></31vailabl>
    <isSearchable><<OBJECT_FIELD_SEARCHABLE_BOOLEAN-VALUE>></isSearchable>
    <isRequired><<OBJECT_FIELD_REQUIRED-OR-NOT_BOOLEAN-VALUE>></isRequired>
    <isReadOnly><<OBJECT_FIELD_EDITABLE-OR-NOT_BOOLEAN-VALUE>></isReadOnly>
    <description><<OBJECT_FIELD_DESCRIPTION-VALUE>></description>
   </item>
   <item>
    <name><<OBJECT_FIELD_EXTERNAL-API_NAME>></name>
    <label><<OBJECT_FIELD_INTERNAL-GUI_NAME>></label>
    <isCustom><<OBJECT_FIELD_CUSTOM-OR-STANDARD_BOOLEAN-VALUE>></isCustom>
    <31vailabl><<OBJECT_FIELD_FIELD-TYPE>></31vailabl>
    <isSearchable><<OBJECT_FIELD_SEARCHABLE_BOOLEAN-VALUE>></isSearchable>
    <isRequired><<OBJECT_FIELD_REQUIRED-OR-NOT_BOOLEAN-VALUE>></isRequired>
    <isReadOnly><<OBJECT_FIELD_EDITABLE-OR-NOT_BOOLEAN-VALUE>></isReadOnly>
    <description><<OBJECT_FIELD_DESCRIPTION-VALUE>></description>
   </item>
  </fields>
 </response>
 <status>
  <detail/>
  <success>true</success>
 </status>
</root>
```

TBE JSON Hyperschema

Method:	Request:
GET	**<<HOST_URL>>**/object/**<<OBJECT_LABEL_NAME>>**/hyperschema.json
	Response:

```
<root>
{
 "title": "<OBJECT_ENTITY_NAME >",
 "description": "<OBJECT_FIELD_DESCRIPTION-VALUE>",
 "allOf": [
  {
   "$ref": "#/definitions/<OBJECT_ENTITY_NAME> "
  }
 ],
 "definitions": {
  "<OBJECT_ENTITY_NAME> ": {
   "allOf": [
    {
     "links": [
      {
       "method": "<ACTION>",
       "title": "<ACTION><OBJECT_ENTITY_NAME> ",
       "description": "<ACTION><OBJECT_ENTITY_NAME>",
       "href": "/object/<OBJECT_ENTITY_NAME> ",
       "rel": "<ACTION>--<OBJECT_ENTITY_NAME> ",
       "targetSchema": {
        "$ref": "#/definitions/<OBJECT_ENTITY_NAME>-response-def"
       },
       "schema": {
        "$ref": "#/definitions/<OBJECT_ENTITY_NAME>-writable-def"
       },
       "templated": true
      },
      {
```

Discovery of Display Fields (for Picklists)

The Get Display Fields request provides the ability to programmatically receive all the data associated with a specific field. This is most beneficial when the field is a picklist or multipicklist of available options that can be assigned to a record. The getMetadata call announces what field types fields are in the 'dataType' output **<<OBJECT_FIELD_FIELD-TYPE>>**.

Taleo Business Edition allows separate application display values versus API values for picklists and multi-picklists. These are defined in the getDisplayField call as displayName versus integrationCode values. For usage with the API, if an integrationCode has been provided, then that is the value expected otherwise when NULL then usage of the displayName is acceptable.

The display fields request also works as a Put method where sending back values will update the display field. When updating picklist values as part of the Put displayField call, the system will automatically deactivate picklist items not sent through on the Put call if previously was available. Deactivate function allows historic reporting and viewing, however does not allow a (application or api) user to select the picklist item again.

There is no JSON hyperschema support for Display Fields.

JSON

Method:	Request:
GET	**<<HOST_URL>>**/object/displayfield/**<<OBJECT_ENTITY_CODE>>**/**<<OBJECT_FIELD_EXTERNAL-API_NAME>>**

	Response:
	
	{
	“response”:{
	“displayField”:{
	“**<<OBJECT_FIELD_EXTERNAL-API_NAME>>**”:{
	“description”:”**<<OBJECT_FIELD_DESCRIPTION-VALUE>>**”,
	“maxLength”:**<<OBJECT_FIELD_MAX-LENGTH>>**,
	“entityType”:”**<<OBJECT_ENTITY_CODE>>**”,
	“33vailabl”:”**<<OBJECT_FIELD_FIELD-TYPE>>**”,
	“fieldType”:”**<<OBJECT_FIELD_FIELD-TYPE-CODE>>**”,
	“externalName”:”**<<OBJECT_FIELD_EXTERNAL-API_NAME>>**”,
	“displayFieldName”:”**<<OBJECT_FIELD_INTERNAL-GUI_NAME>>**”,
	“sortedAscending”:**<<OBJECT_FIELD_PICKLIST_SORT-BOOLEAN-VALUE>>**,
	“selectFirstAsDefault”:**<<OBJECT_FIELD_PICKLIST_DEFAULT-FIRST-BOOLEAN-VALUE>>**,
	“lookupValues”:[
	{
	“displayName”:”**<<OBJECT_FIELD_PICKLIST_INTERNAL_GUI_NAME>>**”,
	“integrationCode”:”**<<OBJECT_FIELD_PICKLIST_EXTERNAL_API_NAME>>**”
	}
	]
	}
	}
	},
	“status”:{
	“detail”:{
	
	},
	“success”:true
	}
	}

XML

Method:	Request:
GET	**<<HOST_URL>>**/object/displayfield/**<<OBJECT_ENTITY_CODE>>**/**<<OBJECT_FIELD_EXTERNAL-API_NAME>>**.xml
	Response: <root> <response> <displayField> <state> <entityType>**<<OBJECT_ENTITY_CODE>>**</entityType> <displayFieldName>**<<OBJECT_FIELD_INTERNAL-GUI_NAME>>**</displayFieldName> <externalName>**<<OBJECT_FIELD_EXTERNAL-API_NAME>>**</externalName> <33vailabl>**<<OBJECT_FIELD_FIELD-TYPE>>**</33vailabl> <description>**<<OBJECT_FIELD_DESCRIPTION-VALUE>>**</description> <maxLength>**<<OBJECT_FIELD_MAX-LENGTH>>**</maxLength> <sortedAscending>**<<OBJECT_FIELD_PICKLIST_SORT-BOOLEAN-VALUE>>**</sortedAscending> <selectFirstAsDefault>**<<OBJECT_FIELD_PICKLIST_DEFAULT-FIRST-BOOLEAN-VALUE>>**</selectFirstAsDefault> <fieldType>**<<OBJECT_FIELD_FIELD-TYPE-CODE>>**</fieldType>

```
      <lookupValues>
       <item>
        <displayName><<OBJECT_FIELD_PICKLIST_INTERNAL_GUI_NAME>></displayName>
        <integrationCode><<OBJECT_FIELD_PICKLIST_EXTERNAL_API_NAME>></integrationCode>
       </item>
       <item>
        <displayName><<OBJECT_FIELD_PICKLIST_INTERNAL_GUI_NAME>></displayName>
        <integrationCode><<OBJECT_FIELD_PICKLIST_EXTERNAL_API_NAME>></integrationCode>
       </item>
      </lookupValues>
     </state>
    </displayField>
   </response>
   <status>
    <detail/>
    <success>true</success>
   </status>
</root>
```

Discovery of Status Values (for Individual Record Assignment)

The Get Status Values request provides the ability to programmatically receive all the status options that can be associated with an entity record.

To find the statuses available per entity, use the object/status/<<OBJECT_ENTITY_CODE>> URL. For a listing of entity codes, refer to the “All API Supported System Entities” table in the “Entity Overview” section of this guide.

For example, if building an API that imports new employees into a Taleo Business Edition instance, you may want to first find what the applicable status ID value is to assign that employee to on create. To do so, you would simply <<HOST_URL>>/object/status/EMPL

JSON

Method:	Request:
GET	**<<HOST_URL>>**/object/status/**<<OBJECT_ENTITY_CODE>>**

Response:

```
{
  “response”:{
    “statuses”:[
      {
        “entityType”:”<<OBJECT_ENTITY_CODE>>”,
        “id”:<<STATUS_ID>>,
        “text”:”<<STATUS_NAME>>”
      },
      {
        “entityType”:”<<OBJECT_ENTITY_CODE>>”,
        “id”:<<STATUS_ID>>,
        “text”:”<<STATUS_NAME>>”
      }
    ]
  },
  “status”:{
    “detail”:{
```

```
    },
    "success":true
  }
}
```

XML

Method:	Request:
GET	**<<HOST_URL>>**/object/status/**<<OBJECT_ENTITY_CODE>>**.xml
	Response:

```
<root>
 <response>
  <statuses>
   <item>
    <entityType><<OBJECT_ENTITY_CODE>></entityType>
    <id><<STATUS_ID>></id>
    <text><<STATUS_NAME>></text>
    <relationshipUrls/>
   </item>
  </statuses>
 </response>
 <status>
  <detail/>
  <success>true</success>
 </status>
</root>
```

This concludes the Introduction and Getting Started portion of the API guide. The next sections dig deeper into all of the API functions and sample code to assist you in development. Please Note: Taleo Business Edition fields can be added, updated and deleted – the sample code provided below should just be used as a reference when developing. The naming of your Taleo fields and available services can be found through the discovery process we just completed.

WEB SERVICE REFERENCE: DISPLAY FIELD ADMINISTRATION

Display Fields request provides the ability to programmatically receive and update all the data associated with a specific field. This is most beneficial when the field is a picklist or multipicklist that may need updating through the API.

Please Note: Display fields are those fields within the system that are exposed to the user within the Administration panel. Geo-Org fields are not display fields, however primary system entities are, these include Candidates, Requisitions, Employees, etc. Please refer to your Taleo Business Edition instance for more details.

Display Field Actions

Please note the following supported RESTful actions:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Display Field Resource		✔	×	×	×	✔	×
description	Text Area	Y				N	
entityType	Text	Y				N	
dataType	Text	Y				N	
fieldType	Picklist	Y				N	
maxLength	Integer	Y				N	
externalName	Text	Y				N	
displayFieldName	Text	Y				Y	
sortedAscending	Boolean	Y				Y	
selectFirstAsDefault	Boolean	Y				Y	
lookUps	Array	Y				Y	

Sample Code Display Fields

Sample code below references a display field that is a picklist; however the URL structure and available actions are the same for non-picklists with the exception of the picklist option lookup.

Please Note: By default the API outputs and assumes JSON. If XML is requested, please ensure adding the attribute of .xml to the end of the URL's in your code, including in the samples provided below. When sending back updates to the picklist items, the system will automatically deactivate items if not returned as part of the update call.

Method	Action	Sample Request
GET	Get Display Field by Name	…/object/displayfield/CAND/state

PUT	Update Display Field by Name	…/object/displayfield/CAND/state
		
		{
		“displayField”:{
		“description”:”Candidate State Selections”,
		“entityType”:”CAND”,
		“externalName”:”state”,
		“displayFieldName”:”State/Territory”,
		“sortedAscending”:true,
		“selectFirstAsDefault”:false,
		“lookupValues”:[
		{
		“displayName”:”US-AK”,

		"integrationCode":"AK"
		},
		{
		"displayName":"US-AL",
		"integrationCode":"AL"
		},
		{
		"displayName":"US-AR",
		"integrationCode":"AR"
		}
		]
		}
		}
		
		OR for XML
		
		<root>
		<displayField>
		<entityType>CAND</entityType>
		<displayFieldName>State/Territory</displayFieldName>
		<externalName>state</externalName>
		<description>xxxx</description>
		<sortedAscending>true</sortedAscending>
		<selectFirstAsDefault>false</selectFirstAsDefault>
		<lookupValues>
		<item>
		<displayName>US-AK</displayName>
		<integrationCode>AR</integrationCode>
		</item>
		</lookupValues>
		</displayField>
		</root>

WEB SERVICE REFERENCE: COMMENTS MANAGEMENT

Primary entities in Taleo Business Edition allow for logging of comments under a comment table. This table is available through the API as well for programmatic access.

For retrieving all comments within an associated entity record, use the following URLs:

<<HOST_URL>>/object/comment/{id}

For retrieving all comments within an associated entity record, use the following URLs:

<<HOST_URL>>/object/**candidate**/{id}/comment

<<HOST_URL>>/object/**requisition**/{id}/comment

<<HOST_URL>>/object/**account**/{id}/comment

<<HOST_URL>>/object/**contact**/{id}/comment

<<HOST_URL>>/object/**employee**/{id}/comment

<<HOST_URL>>/object/**performancereview**/{id}/comment

For retrieving all comments object details, use the following URL

<<HOST_URL>>/object/info/comment

For field definition, use the following URL (please note that custom and standard URL will output the same fields as this is an enclosed table):

<<HOST_URL>>/object/**comment**/description/standard

<<HOST_URL>>/object/**comment**/description/custom

For hyperschema definition, use the following UR

<<HOST_URL>>/object/**comment**/hyperschema

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Comments		✔	✔	×	✔	✔	✔
commentId	Integer	Y	Y		N	N	
creationDate	Date	Y	Y		N	N	
lastUpdated	Date	Y	Y		N	N	
status	Picklist	Y	Y		N	N	
text	String (250)	Y	Y		Y	Y	
entityType	Entity Code	Y	Y		Y	Y	
entityId	Entity Id	Y	Y		Y	Y	

Sample Code Comments

Method	Action	Sample Request
GET	Get Single Comment By ID	…/object/comment/132
POST	Create Comment	…/object/comment/ {“comment”:{“text”:”This is test comment 1…”,”entityType”:”CAND”,”entityId”:74}}
POST	Upsert a comment	…/object/comment?upsert=true {“comment”:{“text”:”This is test comment 1, again…”,”commentId”:45}} **OR XML** <root> <comment> <commentId>117</commentId> <text>This is test comment 1…</text> <entityType>CAND</entityType> <entityId>74</entityId> </comment> </root>
PUT	Update Comment	…/object/comment/13 {“comment”:{“text”:”This is test comment 1, again, and again…”}} **OR for XML** <root> <comment> <text>This is test comment 1…</text>

		</comment> </root>

WEB SERVICE REFERENCE: GEO-ORG ADMINISTRATION

The Taleo Business Edition REST API's that support Administration of geo-org functions are:

- Departments
- Divisions
- Locations – correspond to each geographic location where your company has job openings.
- Regions

Actions will be provided first separated in sub-sections and then sample code showing a department is provided below.

For information on the GEO-ORG functions and their usage within the TBE application, refer to the Administration section in the product help.

Department Actions

Please note the following supported RESTful actions:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Department Resource		✔	✔	×	✔	✔	✔
id	Integer	Y	Y		N	N	Y
creationDate	Date	Y	Y		N	N	
lastUpdated	Date	Y	Y		N	N	
departmentName	Text	Y	Y		**Y***	Y	
departmentCode	Text (50)	Y	Y		**Y**	Y	
description	Text Area	Y	Y		Y	Y	
entityType	Text	Y	Y		N	N	
associatedUsers	Integer	Y	Y		Y	Y	
requisitionApprovers	Integer	Y	Y		Y	Y	

Y* = required for creation.

Relationship URLs:

Relationship	URL
Requisition Approvers	…/object/department/<Id>/requisitionApprovers.xml
Associated Users	…/object/department/<Id>/associatedUsers.xml

Division Actions

Please note the following supported RESTful actions:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Division Resource		✔	✔	×	✔	✔	✔
Id	Integer	Y	Y		N	N	Y

divisionName	Text (100)	Y	Y	**Y***	Y
divisionCode	Text (50)	Y	Y	**Y**	Y
creationDate	Date	Y	Y	N	N
lastUpdated	Date	Y	Y	N	N
Description	Text Area	Y	Y	Y	Y
entityType	Text	Y	Y	Y	Y
associatedUsers	Integer	Y	Y	Y	Y
associatedDepartments	Integer	Y	Y	Y	Y
requisitionApprovers	Integer	Y	Y	Y	Y

Y* = required for creation.

Relationship URLs:

Relationship	URL
Associated Departments	…/object/division/<Id>/associatedDepartments.xml
Requisition Approvers	…/object/division/<Id>/requisitionApprovers.xml
Associated Users	…/object/division/<Id>/associatedUsers.xml

Location Actions

Id used to reference existing locations to update Name and/or Code.

Location name uniqueness is enforced.

Location Code uniqueness is enforced.

Location can be referenced by Location Name only. Location name must be provided if Id is not used.

Please note the following supported RESTful actions:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Location Resource		✔	✔	×	✔	✔	✔
id	Integer	Y	Y		N	N	Y
locationName	Text (100)	Y	Y		Y	Y	
locationCode	Text (50)	Y	Y		Y	Y	
description	Text Area	Y	Y		Y	Y	
phone	Phone	Y	Y		Y	Y	
address	Text	Y	Y		Y	Y	
city	Text	Y	Y		Y	Y	
zipCode	Text	Y	Y		Y	Y	
state	Text	Y	Y		Y	Y	
countryCode	Picklist	Y	Y		Y	Y	
regionId	Integer	Y	Y		Y	Y	
timeZone	Picklist	Y	Y		Y	Y	
directions	Text	Y	Y		Y	Y	
interviewRooms	Text Area	Y	Y		Y	Y	
entityType	Text	Y	Y		Y	N	
requisitionApprovers	Integer	Y	Y		Y	Y	

Relationship URLs:

Relationship	URL

Requisition Approvers	…/object/location/<Id>/requisitionApprovers.xml

Region Actions

Please note the following supported RESTful actions:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Region Resource		✔	✔	×	✔	✔	✔
id	Integer	Y	Y		N	N	Y
regionName	Text(100)	Y	Y		Y	Y	
regionCode	Text(50)	Y	Y		Y	Y	
creationDate	Date	Y	Y		N	N	
lastUpdated	Date	Y	Y		N	N	
description	Text Area	Y	Y		Y	Y	
entityType	Text	Y	Y		N	N	
associatedLocations	Integer	Y	Y		Y	Y	
associatedUsers	Integer	Y	Y		Y	Y	
requisitionApprovers	Integer	Y	Y		Y	Y	

Relationship URLs:

Relationship	URL
Associated Locations	…/object/region/<Id>/associatedLocations.xml
Requisition Approvers	…/object/region/<Id>/requisitionApprovers.xml
Associated Users	…/object/region/<Id>/associatedUsers.xml

Sample Code Geo-Org

Sample code below references departments; however the URL structure and available actions are the same for divisions, locations and regions.

Please Note: By default the API outputs and assumes JSON. If XML is requested, please ensure adding the attribute of .xml to the end of the URL's in your code, including in the samples provided below. For upsert, you will require either passsing the department name or department ID.

Method	Action	Sample Request
GET	Get Department by ID	…/object/department/1
GET	Get All Departments	…/object/department
GET	Get Department by Name	…/object/department?name=Marketing
DELETE	Delete Department by ID	…/object/department/1

POST	Create Department	…/object/department
		
		{
		“department”:{
		“description”:”Sales & Marketing”,
		“departmentName”:”SALES-125”
		}
		}
		
		OR for XML
		
		<root>
		<department>

		<description>Sales & Marketing</description> <departmentName>SALES-125</departmentName> </department> </root>

POST	Upsert Department	…/object/department?upsert=true
		
		{
		"department":{
		"description":"TEST UPSERT",
		"departmentName":"SALES-125"
		}
		}
		
		OR for XML
		
		<root>
		<department>
		<description>TEST UPSERT</description>
		<departmentName>SALES-125</departmentName>
		</department>
		</root>

PUT	Update Department	…/object/department/15 **OR** …/object/department?name=Managers
		
		{
		"department":{
		"departmentName":"Managers",
		"requisitionApprovers":[
		7
		],
		"associatedUsers":[
		7
		],
		"description":"Managers"
		}
		}
		
		OR for XML
		
		<root>
		<department>
		<departmentName>Manager111</departmentName>
		<requisitionApprovers>
		<item>7</item>
		</requisitionApprovers>
		<description>Managers</description>
		<associatedUsers>
		<item>7</item>
		</associatedUsers>
		</department>
		</root>

GET	Retrieves all instances of specified GeoOrg entity	…/object/{GeoOrg entity}/
GET	Search location by name and/or code	…/object/location/(?Name={locationName})(&code={locationCode})

	The same search URL pattern can be applied to region, division and department entities Note: Search on Geo Orgs is exact match	

WEB SERVICE REFERENCE: ENTITY LINK MANAGEMENT

Taleo Business Edition allows for mapping one entity record to another entity record for viewing and reporting purposes within the platform. For example, mapping your (sourcing) candidate account to a requisition; or a (recruiting) candidate contact to a candidate, etc. To replicate via the API, you can use the entitylink API. The entitylink API allows you to provide one entity name (or code) with a record ID and a second entity name (or code) with a record ID and if a relationship exists, they can be linked.

You can manually create relationships for linkage within the application by going to the Administration panel. Please contact your Taleo support representative for details.

To search for a collection of entity links mapped to one specific record, you would send one entity value (name of entity and ID of record):

<<HOST_URL>>/object/entitylink/**<<entityName OR entity Code>>/<<Entity_ID>>**

To search for a specific relationship, send the first entity value and the second entity value:

<<HOST_URL>>/object/entitylink/**<<entityName OR entity Code>>/<<Entity_ID>>/<<entityName OR entity Code>>/<<Entity_ID>>**

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
EntityLink		✔	✔	×	✔	✔	✔
entityType1	Text	Y	Y		**Y***	Y	Y
entityId1	Integer	Y	Y		**Y***	Y	
entityType2	Text	Y	Y		**Y***	Y	
entityId2	Integer	Y	Y		**Y***	Y	

Y* = required for creation.

Sample Code for Entity Links

Method	Action	Sample Request
GET	Get a single relationship link by entity code	…/object/entitylink/ACCT/2/CAND/468
GET	Get a single relationship link by entity name	…/object/entitylink/account/2/candidate/468
GET	Get all Linked records for one record	…/object/entitylink/account/2 OR …/object/entitylink/ACCT/2
DELETE	Delete Linked Relationship	…/object/entitylink/account/2/candidate/468
PUT	Create a Relationship between two separate entity records	…/object/entitylink/account/2/candidate/468

WEB SERVICE REFERENCE: TRANSLATIONS

Please Note: There is no JSON hyperschema support for Translations.

Taleo Business Edition allows for display and data entry in multiple languages of entities. For example, a single requisition can be created and maintained in English, French and Japanese. Users and Candidates can view the Requisition data in the language of their choice.

The Translations endpoint allows for the retrieval of entity data out of TBE in one or more different languages simultaneously. Only GET method is supported.

To retrieve the display value of one or more fields on one or more entities of a certain type in one or more languages:

…/object/translation/<<ENTITY_CODE>?language=<Language(s) code>>&entityId=<<object id(s)&fieldName=<<Fields to retrieve>>

All parameters are required.

Prerequisites:

- You must know the id of the object(s) you wish to retrieve.
- You must know the Entity Code of the object you want to retrieve (REQU for requisition)
- You must know the label of the fields you want to retrieve.

Note:

- If a language specific value does not exists for a given field, the value from the base language on the company is returned.
- Can retrieve only one Entity Type at a time.
- Can retrieve for a list of Languages
- Can retrieve for a list of Object IDs
- Can retrieve for a list of fields

Field Name (GUI)	Field Type	Get	GeT All	Search	Create/)Upsert	Update	Delete
EntityLinK		✔	×	×	×	×	×
Text	Text	Y					
entityId	Integer	Y					
Language	Text	Y					
Fieldname	Text	Y					
entityType	Text	Y					

Sample Code for Translations

Method	Action	Sample Request
GET	Get a values for several fields in several languages from a Requisition.	…/object/translation/REQU.xml?language=fr,en,zh&entityId=61&45vailable=title,status <root> <response> <translations> <item> <translation> <text>Ingénieur</text> <entityId>61</entityId> <language>fr</language> <45vailable>title</45vailable> <entityType>REQU</entityType> </translation> </item> <item> <translation> <text>Ouvert</text> <entityId>61</entityId> <language>fr</language> <45vailable>status</45vailable> <entityType>REQU</entityType> </translation> </item> <item> <translation> <text>Engineer</text> <entityId>61</entityId> <language>en</language> <45vailable>title</45vailable> <entityType>REQU</entityType> </translation> </item> <item> <translation> <text>Open</text> <entityId>61</entityId> <language>en</language> <45vailable>status</45vailable> <entityType>REQU</entityType> </translation> </item> <item> <translation> <text>工程</text> <entityId>61</entityId> <language>zh</language> <45vailable>title</45vailable>

		<entityType>REQU</entityType> </translation> </item> <item> <translation> <text>打开</text> <language>zh</language> <46vailable>status</46vailable> <entityType>REQU</entityType> </translation> </item> </translations> </response> <status> <detail/> <success>true</success> </status> </root>

GET	Get a values for several fields in several languages from a requisition.	…/object/translation/REQU?language=fr,en,zh&entityId=61&47vailable=title,status
		
		{
		“response”:
		{
		“translations”:
		[
		{
		“translation”:
		{
		“text”: “Ingénieur”,
		“entityId”: “61”,
		“language”: “fr”,
		“47vailable”: “title”,
		“entityType”: “REQU”
		}
		},
		{
		“translation”:
		{
		“text”: “Ouvert”,
		“entityId”: “61”,
		“language”: “fr”,
		“47vailable”: “status”,
		“entityType”: “REQU”
		}
		},
		{
		{
		“text”: “Engineer”,
		“entityId”: “61”,
		“language”: “en”,
		“47vailable”: “title”,
		“entityType”: “REQU”
		}
		},
		{
		“translation”:
		{
		“text”: “Open”,
		“entityId”: “61”,
		“language”: “en”,
		“47vailable”: “status”,
		“entityType”: “REQU”
		}
		},
		{

```
        "translation":
        {
          "text": "工程区",
          "entityId": "61",
          "language": "zh",
          "48vailable": "title",
          "entityType": "REQU"
        }
      },
      {
        "translation":
        {
          "text": ""打开",",
          "entityId": "61",
           "language": "zh",
          "48vailable": "status",
          "entityType": "REQU"
        }
      }
    ]
  },
  "status":
  {

    "detail":
    {
    },
    "success": true
  }
}
```

WEB SERVICE REFERENCE: USER ADMINISTRATION

Please note the following supported RESTful actions for Users in Taleo Business Edition:

Please Note: For Searching of User, all Integer, Date and Currency fields should include Parameters <fieldName>_from and/or <fieldName>_to. Please refer to the URL Structure section of this document for full list of parameters and other notes on searching.

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Users		✔	✔	✔	✔	✔	✔
userId	Integer	Y		Y	N	N	Y
creationDate	Date	Y		Y	N	N	
addedWithin	Picklist	N		Y	N	N	
adminNewsletter	Boolean	Y		Y	Y	Y	
phoneAll	Phone	N		N	N	N	
assignReviewAccess	Boolean	Y		Y	Y	Y	
resume	File Upload	N		N	N	N	
submittedEmail	Boolean	Y		Y	Y	Y	
mainStatusContrAccss	Boolean	Y		Y	Y	Y	
compApprover	Boolean	Y		Y	Y	Y	
compContributor	Boolean	Y		Y	Y	Y	
compManager	Boolean	Y		Y	Y	Y	
confidentialAccess	Boolean	Y		Y	Y	Y	
compProxyApprover	User	Y		Y	Y	Y	
newTmplAccess	Boolean	Y		Y	Y	Y	
newRevTempAccess	Boolean	Y		Y	Y	Y	
deleteRevTempAccess	Boolean	Y		Y	Y	Y	
department	Picklist (multiple)	Y		Y	Y	Y	
division	Picklist (multiple)	Y		Y	Y	Y	
editReviewAccess	Boolean	Y	`	Y	Y	Y	
editRevTempAccess	Boolean	Y		Y	Y	Y	
email	Email	Y		Y	Y	Y	
employee	Employee	Y		Y	Y	Y	
fax	Phone	Y		Y	Y	Y	
firstName	Text(30)	Y		Y	Y	Y	
GeneralNewsletter	Boolean	Y		Y	Y	Y	
goalProxyApprover	User	Y		Y	Y	Y	
jobLibAccess	Boolean	Y		Y	Y	Y	
lastLogin	Date	Y		Y	N	N	
lastName	Text(30)	Y		Y	Y	Y	
lastUpdated	Date	Y		Y	N	N	
locale	Picklist	Y		N	Y	Y	
location	Picklist	Y		Y	Y	Y	
manager	Picklist	Y		Y	Y	Y	
middleInitial	Text(30)	Y		Y	Y	Y	
49vailable	Phone	Y		Y	Y	Y	
newReqAccess	Boolean	Y		Y	Y	Y	
offerAppr	Boolean	Y		Y	Y	Y	

offerProxyApprover	User	Y	Y	Y	Y
reviewApprover	Boolean	Y	Y	Y	Y
reviewManager	Boolean	Y	Y	Y	Y
phone	Phone	Y	Y	Y	Y
profile	Text Area	N	N	N	N
region	Picklist (multiple)	Y	Y	Y	Y
approver	Boolean	Y	Y	Y	Y
reqProxyApprover	User	Y	Y	Y	Y
reviewProxyApprover	User	Y	Y	Y	Y
keywords2	Text	N	N	N	N
role	Picklist	Y	Y	Y	Y
status	Picklist	Y	Y	Y	Y
timeZone	Picklist	Y	Y	Y	Y
title	Text(50)	Y	Y	Y	Y
updatedWithin	Picklist	N	Y	N	N
loginName	Text(30)	Y	Y	Y	Y
viewReviewAccess	Boolean	Y	Y	Y	Y
***** CUSTOM FIELDS**		Y	Y	Y	Y

Relationship URLs:

Relationship	URL
Departments	…/object/user/<id>/department
Divisions	…/object/user/<Id/division
Employee	…/object/user/<Id>/employee
Location	…/object/user/<Id>/location
Manager	…/object/user/<Id>/manager
Regions	…/object/user/<Id>/region
Status	…/object/user/<Id>/status
Attachments	…/object/user/<Id>/attachment
History Log	…/object/user/<Id>/historylog

Sample Code Users

Please Note: By default the API outputs and assumes JSON. If XML is requested, please ensure adding the attribute of .xml to the end of the URL's in your request, including in the samples provided below.

For upsert parameter used to conduct a record update, even though you do not need to state the ID in the request URL, you will need to provide the userID in the message body for update to match an existing record.

For update you can only send back the specific fields you are interested in updating.

Method	Action	Sample Request
GET	Get User by ID	…/object/user/1
GET	Search for User	…/object/user/search?start=1&limit=1&addedWithin=5000&lastLogin_from=2008-12-02&lastLogin_to=2008-12-03&offerAppr=false&fields=status|firstName
DELETE	Delete User by ID	…/object/user/1

POST	Create User	…/object/user
		
		{
		“user”:{
		“loginName”:”time1323813641”,
		“lastName”:”webapikztfcnvgqzbdhnyfryjx”,
		“role”:”Administrator”,
		“status”:1,
		“email”:”time1323813641@tberegressionautomation.com”
		}
		}
		
		OR for XML
		
		<root>
		<user>
		<loginName>time1323813641</loginName>
		<lastName>webapibrrwafrtembkhfdfndng</lastName>
		<role>Administrator</role>
		<status type=”integer”>1</status>
		<email>time1323813641@tberegressionautomation.com</email>
		</user>
		</root>

POST	Upsert User	…/object/user?upsert=true
		
		{
		“user”:{
		“loginName”:”time1323975161”,
		“lastName”:”webapihjgazdpacskgkcesmchq”,
		“role”:”Administrator”,
		“userId”:568,
		“status”:1,
		“email”:”time1323975161@tbeautomation.com”
		}
		}
		
		OR for XML
		
		<root>
		<user>
		<userId>568</userId>
		<loginName>UPSERTtest</loginName>
		<lastName>upsertTEST</lastName>
		<role>Administrator</role>
		<status type=”integer”>2</status>
		<email>time1323813641@tasdfberegressionautomation.com</email>
		</user>
		</root>

PUT	Update User	…/object/user/568
		
		{
		“user”:{
		“AdminNewsletter”:true,
		“assignReviewAccess”:true,
		“submittedEmail”:true,
		“mainStatusContrAccss”:false,
		“compApprover”:true,
		“compContributor”:false,
		“compManager”:true,
		“confidentialAccess”:true,
		“newTmplAccess”:true,
		“newRevTempAccess”:false,
		“deleteRevTempAccess”:true,
		“editReviewAccess”:true,
		“editRevTempAccess”:true,
		“email”:”iguz123@taleo.com”,
		“firstName”:”igor1”,
		“GeneralNewsletter”:false,
		“jobLibAccess”:true,
		“lastName”:”guz”,
		“lastUpdated”:”9/29/11 10:25 AM”,
		“locale”:”enUS”,
		“location”:1,
		“manager”:11,
		“newReqAccess”:false,
		“offerAppr”:true,
		“reviewApprover”:false,
		“reviewManager”:true,
		“role”:”Administrator”,
		“status”:1,
		“timeZone”:”America/Los_Angeles”,
		“title”:”TEST”,
		“loginName”:”iguasdf1”,
		“viewReviewAccess”:true,
		}
		}
		
		OR for XML
		
		<root>
		<user>
		<AdminNewsletter>false</AdminNewsletter>
		<assignReviewAccess>true</assignReviewAccess>
		<mainStatusContrAccss>true</mainStatusContrAccss>
		<compApprover>true</compApprover>
		<compContributor>false</compContributor>
		<compManager>true</compManager>
		<confidentialAccess>true</confidentialAccess>
		<newTmplAccess>true</newTmplAccess>
		<newRevTempAccess>true</newRevTempAccess>
		<editPositionAccess>false</editPositionAccess>
		<editProfileAccess>true</editProfileAccess>
		<deleteRevTempAccess>true</deleteRevTempAccess>
		<editReviewAccess>true</editReviewAccess>
		<editRevTempAccess>true</editRevTempAccess>
		<newReqAccess>true</newReqAccess>
		<offerAppr>false</offerAppr>
		<reviewApprover>false</reviewApprover>

		<reviewManager>true</reviewManager> <approver>false</approver> <division> <item>49</item> </division> <department> <item>29</item> </department> <region> <item>1</item> </region> <email>iguz12asdfsad3@taleo.com</email> <employee>27</employee> <fax>23423542345</fax> <firstName>igor1</firstName> <lastName>guz</lastName> <locale>enUS</locale> <location>1</location> <manager>11</manager> <middleInitial></middleInitial> <53vailable></53vailable> <role>Administrator</role> <status>1</status> <timeZone>America/Los_Angeles</timeZone> <title></title> <loginName>iguasasdfasdf1</loginName> <viewReviewAccess>true</viewReviewAccess> <viewPositionAccess>false</viewPositionAccess> <viewProfileAccess>true</viewProfileAccess> </user> </root>

WEB SERVICE REFERENCE: EMPLOYEE ADMINISTRATION

Please note the following supported RESTful actions for employees:

Please Note: For Searching of User, all Integer, Date and Currency fields should include Parameters <fieldName>_from and/or <fieldName>_to. Please refer to the URL Structure section of this document for full list of parameters and other notes on searching.

Please Note: For export of State code, when both Name and Code are used and seperated by a Pipe (I) symbol, the “Use State Code on REST API Exports of Employees” checkbox option on the Employee Picklist Edit: State/Territory page will determine if the two character state code is exported or the state name.

Encrypted fields are only supported with GET if the API User has ‘Confidential Data Access’ (confidentialAccess) defined as true.

Employee work history, residence history, references, education and licenses/certification data is retrievable via the Employee Rolling Entities calls.

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Employee		✔	×	✔	✔	✔	✔
employeeId	Integer	Y		Y	N	N	Y (Delete by ID)
emplAcceptedOfferId	Offer	Y		N	N	N	
ssoLoginOnly	Check Box	Y		Y	Y	Y	
creationDate	Date	Y		Y	N	N	
address	Text(100)	Y		Y	Y	Y	
Address2	Text(100)	Y		Y	Y	Y	
areasOfExpertise	Text Area	Y		Y	Y	Y	
candidate	Custom	Y		Y	N	N	
careerAmbitions	Text Area	Y		Y	Y	Y	
city	Text(30)	Y		Y	Y	Y	
country	Picklist	Y		Y	Y	Y	
criticalExperience	Text Area	Y		Y	Y	Y	
criticalPosition	Check Box	Y		Y	Y	Y	
birthdate	Encrypted Date * If user has confidential access	Y		N	N	N	
department	Picklist	Y		Y	Y	Y	
desiredAdvancement	Picklist	Y		Y	Y	Y	
developmentPath	Picklist (multiple)	Y		Y	Y	Y	
division	Picklist (multiple)	Y		Y	Y	Y	
email	Email	Y		Y	Y	Y	
numberOfReports	Integer	Y		N	Y	Y	
numberOfSubordinates	Integer	Y		N	Y	Y	
employeeNumber	Text(75)	Y		Y	Y	Y	
emplIsActiveStatus	Check Box	Y		Y	Y	Y	
lastReviewsScoreAvg	Custom	Y		N	N	N	
lastReviewManager	User	Y		N	N	N	
lastReviewScore	Custom	Y		N	N	N	
lockedFromEws	Check Box	Y		Y	Y	Y	
emplUserIsReviewMgr	Check Box	Y		Y	Y	Y	
ewsLogin	Text(75)	Y		Y	Y	Y	
ewsPassword	Password	N		N	Y	Y	
duration	Picklist	Y		Y	Y	Y	
firstName	Text(30)	Y		Y	Y	Y	
flightRiskReason	Text	Y		Y	Y	Y	
flightRisk	Check Box	Y		Y	Y	Y	
gender	Picklist	Y		Y	Y	Y	
highPerformer	Check Box	Y		Y	Y	Y	
highestEducationLev	Picklist	Y		Y	Y	Y	
hiredDate	Date	Y		Y	Y	Y	
hiredForReqId	Integer	Y		Y	Y	Y	
hiredForReqJobCode	Text(20)	Y		Y	Y	Y	
hiredForReqTitle	Text(150)	Y		Y	Y	Y	
hourlyWage	Currency	Y		Y	Y	Y	
reportsIndirectly	Employee	Y		Y	Y	Y	
IndividualWithDisability	Picklist	Y		Y	Y	Y	
activeEmplPosition	Check Box	Y		Y	N	N	
jobClassification	Picklist	Y		Y	Y	Y	

jobCode	Text(50)	Y	Y	Y	Y
jobTitle	Text(50)	Y	Y	Y	Y
languagesSpoken	Picklist (multiple)	Y	Y	Y	Y
lastDayDate	Date	Y	Y	Y	Y
lastName	Text(30)	Y	Y	**Y***	Y
lastUpdated	Date	Y	Y	N	N
leadershipQual	Picklist (multiple)	Y	Y	Y	Y
licenseNumber	Encrypted Text	Y	Y	Y	Y
location	Picklist	Y	Y	Y	Y
manager	Manager	Y	Y	Y	Y
middleInitial	Text(30)	Y	Y	Y	Y
cellphone	Phone	Y	Y	Y	Y
nextLevelPosRank	Picklist	Y	Y	Y	Y
offBoardStatus	Picklist	Y	Y	Y	Y
onBoardStatus	Picklist	Y	Y	Y	Y
emplEmailRequired	Check Box	Y	Y	Y	Y
passportNumber	Encrypted Text	Y	Y	Y	Y
payFrequency	Picklist	Y	Y	Y	Y
phone	Phone	Y	Y	Y	Y
potential	Picklist	Y	Y	Y	Y
preferredLocale	Picklist	Y	Y	Y	Y
preferredLocations	Picklist (multiple)	Y	Y	Y	Y
employeePicture	File Upload	N	N	N	N
promotionReadiness	Picklist	Y	Y	Y	Y
race	Picklist	Y	Y	Y	Y
region	Picklist (multiple)	Y	Y	Y	Y
Religion	Picklist	Y	Y	Y	Y
resignedDate	Date	Y	Y	Y	Y
reviewApprover	User	Y	Y	Y	Y
reviewManager	Review Manager	Y	Y	Y	Y
reviewTemplate	Performance Review Template	Y	Y	Y	Y
salary	Currency	Y	Y	Y	Y
salaryGrade	Text(30)	Y	Y	Y	Y
Salutation	Text(30)	Y	Y	Y	Y
ssn	Encrypted Text * If user has confidential access	Y	N	N	N
startDate	Date	Y	Y	Y	Y
state	Picklist	Y	Y	Y	Y
status	Picklist	Y	Y	**Y***	Y
stockOptions	Text	Y	Y	Y	Y
veteran	Picklist	Y	Y	Y	Y
willingnessToRel	Picklist	Y	Y	Y	Y
zipCode	Text(10)	Y	Y	Y	Y
ebsEmployeeNumber	Text(100) * If enabled in TBE	Y	Y	Y	Y
ebsIntegrationStatus	Picklist * If enabled in TBE	Y	Y	Y	Y
fusionEmployeeNumber	Text(100) * If enabled in TBE	Y	Y	Y	Y
fusionIntegrationStatus	Picklist * If enabled in TBE	Y	Y	Y	Y
jdeEmployeeNumber	Text(100) * If enabled in TBE	Y	Y	Y	Y

jdeIntegrationStatus	Picklist * If enabled in TBE	Y	Y	Y	Y
nsuiteEmployeeNumber	Text(100) * If enabled in TBE	Y	Y	Y	Y
nsuiteIntegrationStatus	Picklist * If enabled in TBE	Y	Y	Y	Y
psoftEmployeeNumber	Text(100) * If enabled in TBE	Y	Y	Y	Y
psoftIntegrationStatus	Picklist * If enabled in TBE	Y	Y	Y	Y
addedWithin	Picklist	N	Y	N	N
phoneAll	Phone	Y	N	N	N
payRange	Pay Range	N	N	N	N
updatedWithin	Picklist	N	Y	N	N
***** CUSTOM FIELDS**		Y	Y	Y	Y

Y* = required for creation.

Relationship URLs:

Relationship	URL
Candiate	…/object/employee/<Id>/candidate
Department	…/object/employee/<Id>/department
Division	…/object/employee/<Id>/division
Reports Indirectly	…/object/employee/Id/reportsIndirectly
Last Review Manager	…/object/employee/Id/lastReviewManager
Location	…/object/employee/<Id>/location
Manager	…/object/employee/<Id>/manager
Off-Board Status	…/object/employee/<Id>/offBoardStatus
On-Board Status	…/object/employee/<Id>/onBoardStatus
Region	…/object/employee/<Id>/region
Review Manager	…/object/employee/<Id>/reviewManager
Status	…/object/employee/<Id>/status
Work History	…/object/employee/<Id>/workhistory
Reference	…/object/employee/<Id>/reference
Education	…/object/employee/<Id>/education
Residence	…/object/employee/<Id>/residence
Certificate	…/object/employee/<Id>/certificate
Packets	…/object/employee/<Id>/packet
Comment	…/object/employee/<Id>/comment
History Log	…/object/employee/<Id>/historylog
Contact Log	…/object/employee/<Id>/contactlog

Sample Code Employee Record

Sample code below references employee record.

Please Note: By default the API outputs and assumes JSON. If XML is requested, please ensure adding the attribute of .xml to the end of the URL's in your request, including in the samples provided below.

For upsert parameter used to conduct a record update, even though you do not need to state the ID in the request URL, you will need to provide the employeeNumber or employeeID in the message body for update to match an existing record.

For update you can only send back the specific fields you are interested in updating.

ewsPassword – Employee Website Password field will come back masked in responses for security reasons. You may overwrite an ewsPassword by including it in your PUT/POST call, however it will always be masked following (in GET call).

Method	Action	Sample Request
GET	Get Employee by ID	…/object/employee/1
GET	Search for Employee	…/object/employee/search?start=1&limit=1&addedWithin=5000&lastLogin_from=2008-12-02&lastLogin_to=2008-12-03&offerAppr=false&fields=status|firstName
DELETE	Delete Employee by ID	…/object/employee/1

POST	Create Employee	…/object/employee
		
		{
		“employee”:{
		“address”:”webapixsaugdxptxqtbgpvwbrq”,
		“city”:”webapiqyjhzpqcsnnqbjyuutpf”,
		“lastName”:”webapizfvngxnjmvbhwrgdayut”,
		“57vailable”:”5112246641”,
		“firstName”:”webapigjhaunnabebaapgumzdd”,
		“phone”:”5834138253”,
		“middleInitial”:”webapifrvzdqcqyrmfqzprhrnh”,
		“status”:1,
		“state”:”US-CA”,
		“email”:”time1323813611@tbeautomation.com”,
		“ewsLogin”:”time1323813611”
		}
		
		OR for XML
		
		<root>
		<employee>
		<division>
		<item>49</item>
		</division>
		<email>email5@eee.fff</email>
		<employeeNumber>EMP000118</employeeNumber>
		<emplIsActiveStatus>false</emplIsActiveStatus>
		<gender>Female</gender>
		<highPerformer>false</highPerformer>
		<highestEducationLev>Degree</highestEducationLev>
		<hiredDate>2008-11-05</hiredDate>
		<hourlyWage>11.00</hourlyWage>
		<activeEmplPosition>false</activeEmplPosition>
		<jobCode>17</jobCode>
		<jobTitle>Nurse</jobTitle>
		<languagesSpoken/>
		<lastReviewsScoreAvg></lastReviewsScoreAvg>
		<lastDayDate>null</lastDayDate>
		<lastReviewManager>null</lastReviewManager>
		<lastReviewScore></lastReviewScore>
		<lastName>Employee9</lastName>
		<location>1</location>
		<middleInitial>B</middleInitial>
		<57vailable>555/555-5555</57vailable>
		<nextLevelPosRank></nextLevelPosRank>

		<offBoardStatus>1</offBoardStatus> <onBoardStatus>3</onBoardStatus> <preferredLocale>enUS</preferredLocale> <race>Asian (not Hispanic or Latino)</race> <region> <item>1</item> <item>38</item> <item>50</item> </region> <status>1</status> </employee> </root>

POST	Upsert Employee	…/object/employee?upsert=true
		
		{
		“employee”:{
		“status”:”5”,
		“employeeId”:15
		}
		}
		
		OR for XML
		
		<root>
		<employee>
		<employeeNumber>EMP000118</employeeNumber>
		<employeeId>15</employeeId>
		<hourlyWage>19.00</hourlyWage>
		</employee>
		</root>

PUT	Update Employee	…/object/employee/15
		
		{
		“employee”:{
		“address”:”webapixsaugdxptxqtbgpvwbrq”,
		“city”:”webapiqyjhzpqcsnnqbjyuutpf”,
		“lastName”:”webapizfvngxnjmvbhwrgdayut”,
		“58vailable”:”5112246641”,
		“firstName”:”webapigjhaunnabebaapgumzdd”,
		“phone”:”5834138253”,
		“middleInitial”:”webapifrvzdqcqyrmfqzprhrnh”,
		“status”:1,
		“state”:”US-CA”,
		“email”:”time1323813611@tbeautomation.com”,
		“ewsLogin”:”time1323813611”
		}
		
		OR for XML
		
		<root>
		<employee>

		<division> <item>49</item> </division> <email>email5@eee.fff</email> <employeeNumber>EMP000118</employeeNumber> <emplIsActiveStatus>false</emplIsActiveStatus> <gender>Female</gender> <highPerformer>false</highPerformer> <highestEducationLev>Degree</highestEducationLev> <hiredDate>2008-11-05</hiredDate> <hourlyWage>11.00</hourlyWage> <activeEmplPosition>false</activeEmplPosition> <jobCode>17</jobCode> <lastName>Employee9</lastName> <location>1</location> <middleInitial>B</middleInitial> <59vailable>555/555-5555</59vailable> <offBoardStatus>1</offBoardStatus> <onBoardStatus>3</onBoardStatus> <preferredLocale>enUS</preferredLocale> <race>Asian (not Hispanic or Latino)</race> <region> <item>1</item> <item>38</item> <item>50</item> </region> <status>1</status> </employee> </root>

WEB SERVICE REFERENCE: EMPLOYEE ROLLING ENTITIES

The Taleo Business Edition employee rolling entities are dynamic forms of historic talent information that is tied to a single employee. These are:

- Work History
- Education
- Licenses & Certifications
- References
- Previous Addresses

There is no search associated with rolling entities, however a GET ALL function that is tied to a specific employee has been provided which is specific to a employee record. To access them, the URL format would be:

<<HOST_URL>>/object/employee/**<<EMPLOYEE_ID>>**/**<<ROLLING_ENTITY_NAME>>**

Where the rolling entity name would be:

- workhistory
- education
- certificate
- reference
- residence

If a single instance is requested, use the rolling entity ID post the URL:

<<HOST_URL>>/object/employee/**<<EMPLOYEE_ID>>**/**<<ROLLING_ENTITY_NAME>>**/**<<ROLLING_ENTITY_ID>>**

The getMetadata call for a classic REST API rolling entity would be for standard fields only vs including custom fields:

<<HOST_URL>>/object/employee/**<<ROLLING_ENTITY_NAME>>**/description/**standard**

<<HOST_URL>>/ object/employee/**<<ROLLING_ENTITY_NAME>>**/description/**custom**

The getMetadata call for a JSON hyperschema rolling entity:

<<HOST_URL>>/object/employee/**<<ROLLING_ENTITY_NAME>>**/hyperschema.json

Work History Actions

Please note the following supported RESTful actions:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Work History Resource		✔	✔	×	✔	✔	✔
id	Integer	Y	Y		Y	Y	Y (Delete by ID)
workHistCityState	Text(100)	Y	Y		Y	Y	
workHistStreet	Text(100)	Y	Y		Y	Y	
workHistCoName	Text(100)	Y	Y		Y	Y	
workHistPhone	Phone	Y	Y		Y	Y	
workHistDesc	Text Area	Y	Y		Y	Y	
workHistSupervisor	Text(100)	Y	Y		Y	Y	
workHistExplanation	Text(100)	Y	Y		Y	Y	
workHistPay	Text(20)	Y	Y		Y	Y	
workHistFrom	Text[1]	Y	Y		Y	Y	
workHistOkContact	Check Box	Y	Y		Y	Y	
workHistReasonForLeaving	Picklist	Y	Y		Y	Y	
workHistSuperTitle	Text(100)	Y	Y		Y	Y	
workHistJobTitle	Text(100)	Y	Y		Y	Y	
workHistTo	Text[1]	Y	Y		Y	Y	
***** CUSTOM FIELDS**		Y	Y		Y	Y	

Notes:

1. **workHistFrom and workHistTo expect a format of 'MM-YYYY', for example '09-2013', or the value 'To Present'. Values provided are validated.**

Education Actions

Please note the following supported RESTful actions:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Education Resource		✔	✔	×	✔	✔	✔
id	Integer	Y	Y		Y	Y	Y (Delete by ID)

eduHistCity	Text	Y	Y	Y	Y
eduHistFrom	Text[1]	Y	Y	Y	Y
eduHistTo	Text[1]	Y	Y	Y	Y
eduHistDegreeAchieved	Text	Y	Y	Y	Y
eduHistFieldOfStudy	Text	Y	Y	Y	Y
eduHistSchool	Text	Y	Y	Y	Y
eduHistState	Picklist	Y	Y	Y	Y
***** CUSTOM FIELDS**		Y	Y	Y	Y

Notes:

1. **eduHistFrom and eduHistTo expect a format of 'MM-YYYY', for example '09-2013', or the value 'To Present'. Values provided are validated.**

Licenses & Certificate Actions

Please note the following supported RESTful actions:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Licenses Resource		✔	✔	×	✔	✔	✔
id	Integer	Y	Y		Y	Y	Y (Delete by ID)
certName	Text	Y	Y		Y	Y	Y
certIssuingBody	Text	Y	Y		Y	Y	Y
certYear	Text(4)	Y	Y		Y	Y	Y
***** CUSTOM FIELDS**		Y	Y		Y	Y	Y

References Actions

Please note the following supported RESTful actions:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
References Resource		✔	✔	×	✔	✔	✔
id	Integer	Y	Y		Y	Y	Y (Delete by ID)
refEmail	Email	Y	Y		Y	Y	
refPhone	Phone	Y	Y		Y	Y	
refName	Text	Y	Y		Y	Y	
refYearsKnown	Text(2)	Y	Y		Y	Y	
***** CUSTOM FIELDS**		Y	Y		Y	Y	

Previous Addresses Actions

Please note the following supported RESTful actions:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Address Resource		✔	✔	×	✔	✔	✔
id	Integer	Y	Y		Y	Y	Y (Delete by ID)
resHistCity	Text	Y	Y		Y	Y	
resHistCountry	Picklist	Y	Y		Y	Y	
resHistFrom	Text[1]	Y	Y		Y	Y	
resHistState	Picklist	Y	Y		Y	Y	
resHistStreet	Text	Y	Y		Y	Y	
resHistTo	Text[1]	Y	Y		Y	Y	
resHistZipCode	Text	Y	Y		Y	Y	
***** CUSTOM FIELDS**		Y	Y		Y	Y	

Notes:

1. **resHistFrom and resHistTo expect a format of 'MM-YYYY', for example '09-2013', or the value 'To Present'. Values provided are validated.**

Sample Code Employee Rolling Entities

Sample code below references a work history instance; however the URL structure and available actions are the same for the other employee rolling entities.

Please Note: By default the API outputs and assumes JSON. If XML is requested, please ensure adding the attribute of .xml to the end of the URL's in your code, including in the samples provided below.

Method	Action	Sample Request
GET	Get single Employee work history by ID	…/object/employee/41/workhistory/2112
GET	Get all work history for a single employee	…/object/employee/41/workhistory/
DELETE	Delete single work history instance	…/object/employee/41/workhistory/2112
POST	Create work history for an employee	…/object/employee/41/workhistory { "workhistory": { "workHistCityState": "US-CA", "workHistCoName": "Taleo", "workHistPhone": "925-483-2342", "workHistStreet": "10 Happy Street", "workHistFrom": "09-1997", "workHistTo": "To Present", "workHistDesc": "Best Manager", "workHistSupervisor": "Jason", "workHistExplanation": "Perfect Manager",

		“workHistPay”: “14.00”, “workHistOkContact”: true, “workHistSuperTitle”: “Manager”, “workHistJobTitle”: “Manager” } } } **OR for XML** <root> <workhistory> <workHistCityState>US-CA</workHistCityState> <workHistCoName>Taleo</workHistCoName> <workHistPhone>925-111-1111</workHistPhone> <workHistStreet>945 Dublin Blvd</workHistStreet> <workHistFrom>09-1997</workHistFrom> <workHistTo>To Present</workHistTo> <workHistDesc>Wrestler</workHistDesc> <workHistSupervisor></workHistSupervisor> <workHistExplanation></workHistExplanation> <workHistPay></workHistPay> <workHistOkContact>false</workHistOkContact> <workHistReasonForLeaving></workHistReasonForLeaving> <workHistSuperTitle></workHistSuperTitle> <workHistJobTitle>sDBV</workHistJobTitle> </workhistory> </root>

POST	Upsert work history for an employee	…/object/employee/41/workhistory?upsert=true
		
		{
		“workhistory”:
		{
		“workHistCityState”: “US-CA”,
		“workHistCoName”: “Taleo”,
		“workHistPhone”: “925-483-2342”,
		“workHistStreet”: “10 Happy Street”,
		“workHistFrom”: “09-1997”,
		“workHistTo”: “To Present”,
		“workHistDesc”: “Best Manager”,
		“workHistSupervisor”: “Jason”,
		“workHistExplanation”: “Perfect Manager”,
		“workHistPay”: “14.00”,
		“workHistOkContact”: true,
		“workHistSuperTitle”: “Manager”,
		“workHistJobTitle”: “Manager”,
		“id”: 2112
		}
		}
		}
		
		OR for XML

		<root> <workhistory> <workHistCityState></workHistCityState> <workHistCoName></workHistCoName> <workHistPhone></workHistPhone> <workHistStreet></workHistStreet> <workHistFrom>09-1997</workHistFrom> <workHistTo>To Present</workHistTo> <workHistDesc></workHistDesc> <workHistSupervisor></workHistSupervisor> <workHistExplanation></workHistExplanation> <workHistPay></workHistPay> <workHistOkContact>false</workHistOkContact> <workHistReasonForLeaving></workHistReasonForLeaving> <workHistSuperTitle></workHistSuperTitle> <workHistJobTitle>sDBV</workHistJobTitle> <id>2112</id> </workhistory> </root>

PUT	Update a field for a work history for an employee	…/object/employee/41/workhistory/2112
		
		{
		“workhistory”:
		{
		“workHistCityState”: “US-CA”,
		“workHistCoName”: “Taleo”,
		“workHistPhone”: “925-483-2342”,
		“workHistStreet”: “10 Happy Street”,
		“workHistFrom”: “09-1997”,
		“workHistTo”: “To Present”,
		“workHistDesc”: “Best Manager”,
		“workHistSupervisor”: “Jason”,
		“workHistExplanation”: “Perfect Manager”,
		“workHistPay”: “14.00”,
		“workHistOkContact”: true,
		“workHistSuperTitle”: “Manager”,
		“workHistJobTitle”: “Manager”,
		“id”: 2112
		}
		}
		}
		
		OR for XML
		
		<root>
		<workhistory>
		<workHistTo>To Present</workHistTo>
		<workHistFrom>09-1997</workHistFrom>
		<id>2112</id>
		</workhistory>
		</root>

WEB SERVICE REFERENCE: EMPLOYEE GOALS

Please note the following supported RESTful actions for employee goals:

Please Note: For Searching of Employee Goals, all Integer, Date and Currency fields should include Parameters <fieldName>_from and/or <fieldName>_to. Please refer to the URL Structure section of this document for full list of parameters and other notes on searching.

Please Note: There is no JSON hyperschema support for Employee Goals.

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Employee Goals Resource		✔	×	✔	✔	✔	✔
goalId	Integer	Y		Y	N	N	Y
corpGoals	Custom	Y		Y	Y	Y	
goalApprovalStatus	Picklist	Y		Y	Y	Y	
completedDate	Date	Y		Y	Y	Y	
createdById	User	Y		Y	Y	Y	
creationDate	Date	Y		Y	N	N	
department	Picklist	Y		Y	N	N	
description	Text Area	Y		Y	Y	Y	
division	Picklist (multiple)	Y		Y	N	N	
dueDate	Date	Y		Y	**Y***	Y	
employeeId	Integer	Y		Y	**Y***	Y	
employeeManager	Manager	Y		Y	Y	Y	
employee	Employee	Y		Y	Y	Y	
employeeAcceptedDate	Date	Y		Y	Y	Y	
employeeComment	Text Area	Y		Y	Y	Y	
employeeScore	Custom	Y		N	N	Y	
lastProgressUpdate	Date	Y		Y	N	N	
lastUpdated	Date	Y		Y	N	N	
location	Picklist	Y		Y	N	N	
managerComment	Text Area	Y		Y	Y	Y	
managerScore	Custom	Y		Y	N	Y	
measurement	Text Area	Y		Y	Y	Y	
isPrivate	Check Box	Y		Y	Y	Y	
region	Picklist (multiple)	Y		Y	N	N	
reviewManager	Review Manager	Y		Y	Y	Y	
status	Picklist	Y		Y	**Y***	Y	
goalTitle	Text(200)	Y		Y	**Y***	Y	
reviewType	Picklist	Y		Y	**Y***	Y	
goalPercent	Integer	Y		Y	Y	Y	
***** CUSTOM FIELDS**		Y		Y	Y	Y	

Y* = required for creation.

Relationship URLs:

Relationship	URL
Goal Approval Status	…/object/employeegoal/<Id>/goalApprovalStatus
Created by ID	…/object/employeegoal/Id/createdById
Department	…/object/employeegoal/<Id>/department
Division	…/object/employeegoal/<Id>/division

Employee Manager	…/object/employeegoal/<Id>/employeeManager
Location	…/object/employeegoal/<Id>/location
Region	…/object/employeegoal/<Id>/region
Review Manager	…/object/employeegoal/<Id>/reviewManager
Status	…/object/employeegoal/<Id>/status
History Log	…/object/employeegoal/<Id>/historylog

Sample Code Employee Goals

Sample code below references employee goals.

Please Note: By default the API outputs and assumes JSON. If XML is requested, please ensure adding the attribute of .xml to the end of the URL's in your request, including in the samples provided below.

For upsert parameter used to conduct a record update, even though you do not need to state the ID in the request URL, you will need to provide the goalID in the message body for update to match an existing record. Otherwise assumption is a new record creation.

Updating of region, division, location and other fields provided on the GET call may not be supported as they are derived directly from the employee being assigned the employee goal.

Method	Action	Sample Request
GET	Get Employee Goal by ID	…/object/employeegoal/1
GET	Search for Employee Goal	…/object/employeegoal/search?start=1&limit=999&employee=14&status=1&lastUpdated_from=2011-12-16&fields=goalTitle|goalPercent|managerScore|managerComment
DELETE	Delete Employee Goal by ID	…/object/employeegoal/1

POST	Create Employee Goal	…/object/employeegoal
		
		{
		"employeegoal":{
		"corpGoals":{
		"isCompanyGoal":true,
		"id":2
		},
		"goalApprovalStatus":1,
		"completedDate":"2011-03-04",
		"createdById":56,
		"description":"Test Automation Goal Description",
		"dueDate":"2011-03-31",
		"employeeId":"14",
		"employeeManager":27,
		"employee":"14",
		"employeeAcceptedDate":"2011-03-31",
		"employeeComment":"Test",
		"measurement":"Test Automation Goal Measurement",
		"isPrivate":false,
		"reviewManager":27,
		"status":4,
		"goalTitle":"Test Automation Goal",
		"reviewType":"Quarterly",
		"goalPercent":"0"
		}
		}

		OR for XML <root> <employeegoal> <corpGoals> <isCompanyGoal>true</isCompanyGoal> <id>2</id> </corpGoals> <goalApprovalStatus>1</goalApprovalStatus> <completedDate>2011-03-04</completedDate> <createdById>44</createdById> <description>Test Automation Goal Description</description> <dueDate>2011-03-31</dueDate> <employeeId>14</employeeId> <employeeManager>27</employeeManager> <employee>14</employee> <employeeAcceptedDate>2011-03-31</employeeAcceptedDate> <employeeComment>Test</employeeComment> <managerComment></managerComment> <managerScore>null</managerScore> <measurement>Test Automation Goal Measurement</measurement> <isPrivate>false</isPrivate> <reviewManager>27</reviewManager> <status>4</status> <goalTitle>Test Automation Goal</goalTitle> <reviewType>Quarterly</reviewType> <goalPercent>0</goalPercent> </employeegoal> </root>

POST	Upsert Employee Goal	…/object/employeegoal?upsert=true
		
		{
		“employeegoal”:{
		“status”:”1”,
		“goalId”:146
		}
		}
		
		OR for XML
		
		<root>
		<employeegoal>
		< goalId>146</employeeId>
		<measurement>Made Goal Private</measurement>
		<isPrivate>true</isPrivate>
		</employeegoal>
		<root>

PUT	Update Employee Goal	…/object/employeegoal/146
		
		{
		“employeegoal”:{

		“status”:”2” } } **OR for XML** <root> <employeegoal> <goalPercent>100</goalPercent> </employeegoal> <root>

WEB SERVICE REFERENCE: PERFORMANCE REVIEWS

Please Note: For Searching of Employee Performance Reviews, all Integer, Date and Currency fields should include Parameters <fieldName>_from and/or <fieldName>_to. Please refer to the URL Structure section of this document for full list of parameters and other notes on searching.

System will not allow you to update or upsert performance reviews that have a status of Final (ID 5).

Please see Appendix A for information about Performance Review attachments.

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Performance Reviews Resource		✔	×	✔	✔	✔	✔
reviewId	Integer	Y		Y	N	N	Y
reviewTemplateId	Integer	N		N	Y	N	
creationDate	Date	Y		Y	N	N	
addedWithin	Picklist	N		Y	N	N	
employeeAverage	Custom	Y		Y	N	N	
managerAverage	Custom	Y		Y	N	N	
contributors	Custom	Y		Y	Y	Y	
currUserApprStatus	Check Box	N		N	N	N	
department	Picklist	Y		Y	N	N	
division	Picklist (multiple)	Y		Y	N	N	
dueDate	Date	Y		Y	Y	Y	
emplCalcAchievement	Decimal	Y		Y	Y	Y	
emplComments	Text Area	Y		Y	Y	Y	
employee	Employee	Y		Y	Y	Y	
emplRating	Custom	Y		Y	Y	Y	
emplSignature	Text(256)	Y		Y	Y	Y	
emplSignatureDate	Date	Y		Y	Y	Y	
emplSubmittedDate	Date	Y		Y	Y	Y	
endDate	Date	Y		Y	Y	Y	
jobTitle	Text(50)	Y		Y	Y	Y	
lastUpdated	Date	Y		Y	N	N	
location	Picklist	Y		Y	N	N	
mgrCalcAchievement	Decimal	Y		Y	Y	Y	
mgrComments	Text Area	Y		N	Y	Y	
mgrEditFormVer	Custom	N		N	N	N	
mgrRating	Custom	Y		Y	Y	Y	

mgrSignature	Text(256)	Y	Y	Y	Y
mgrSignatureDate	Date	Y	Y	Y	Y
maxScore	Decimal	Y	Y	Y	Y
multiRaterDueDate	Date	Y	Y	Y	Y
multiRaterStatus	Picklist	Y	Y	Y	Y
approvers	Custom	Y	Y	Y	Y
region	Picklist (multiple)	Y	Y	N	N
reviewCode	Text(50)	Y	Y	Y	Y
finalizedDate	Date	Y	Y	Y	Y
reviewManager	User	Y	Y	Y	Y
reviewType	Picklist	Y	Y	Y	Y
startDate	Date	Y	Y	Y	Y
status	Picklist	Y	Y	Y	Y
title	Text(256)	Y	Y	Y	Y
updatedWithin	Picklist	N	Y	N	N
useGoalCalculation	Check Box	Y	Y	Y	Y
workflowDefId	Picklist	Y	Y	Y	Y
***** CUSTOM FIELDS**		N	Y	N	N

Relationship URLs:

Relationship	URL
Department	…/object/pmreview/<reviewId>/department
Division	…/object/pmreview/<reviewId>/division
Employee	…/object/pmreview/<reviewId>/employee
Location	…/object/pmreview/<reviewId>/location
Multi Rater Status	…/object/pmreview/<reviewId>/multiRaterStatus
Approvers	…/object/pmreview/<reviewId>/approvers
Contributors	…/object/pmreview/<reviewId>/contributors
Review Manager	…/object/pmreview/<reviewId>/reviewManager
Status	…/object/pmreview/<reviewId>/status
Comment	…/object/pmreview/<reviewId>/comment
History Log	…/object/pmreview/<reviewId>/historylog

Sample Code Performance Reviews

Please Note: By default the API outputs and assumes JSON. If XML is requested, please ensure adding the attribute of .xml to the end of the URL's in your request, including in the samples provided below.

For update you can only send back the specific fields you are interested in updating. Upsert and update calls cannot be conducted on finalized performance reviews. For information on attachments please see Appendix A.
Updating of region, division, location and other fields provided on the GET call may not be supported as they are derived directly from the employee being assigned the performance review and available for get and search functions only.

A reviewTemplateId needs to be assigned when creating a new performance review. Review Template ID's are retrievable from the application.

Method	Action	Sample Request
GET	Get PM Review by ID	…/object/pmreview/1
GET	Search for PM Review	…/object/pmreview/search?start=1&limit=1&finalizedDate_from=2010-03-03
DELETE	Delete PM Review	…/object/pmreview/1
POST	Create PM Review	…/object/pmreview

		{
		"pmreview":{
		"dueDate":"2011-03-11",
		"emplCalcAchievement":"Sample rating review",
		"emplComments":"test",
		"employeeDueDate":"2011-03-11",
		"employee":14,
		"emplRating":3.9,
		"emplSignature":"Employee",
		"emplSignatureDate":"2011-03-12",
		"emplSubmittedDate":"2011-03-12",
		"endDate":"2011-03-12",
		"IG":"asdf",
		"jobTitle":"Nurse",
		"mgrCalcAchievement":"Rating Final",
		"mgrComments":"test",
		"mgrRating":3.5,
		"mgrSignature":"Manager",
		"mgrSignatureDate":"2011-03-04",
		"reviewCode":"RTRC",
		"finalizedDate":"2011-12-02",
		"reviewManager":44,
		"reviewType":"Quarterly",
		"startDate":"2011-03-04",
		"status":1,
		"title":"Test Review Template",
		"useGoalCalculation":false,
		"workflowDefId":29,
		"reviewTemplateId":106
		}
		}
		
		OR for XML
		
		<root>
		<pmreview>
		<employee>14</employee>
		<reviewManager>44</reviewManager>
		<reviewTemplateId>106</reviewTemplateId>
		<reviewType>Quarterly</reviewType>
		<startDate>2011-03-04</startDate>
		<status>1</status>
		<title>Test Review Template</title>
		</pmreview>
		</root>

PUT	Update PM Review	…/object/pmreview/146
		
		{
		"pmreview":{
		"status":"1"
		}
		}

		OR for XML <root> <pmreview> <reviewManager>54</reviewManager> <startDate>2011-03-04</startDate> <status>1</status> </pmreview> </root>

WEB SERVICE REFERENCE: COMPANY GOALS

Please Note: There is no JSON hyperschema support for Employee Goals.

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Company Goals Resource		✔	✔	×	✔	✔	✔
corpGoalId	Integer	Y	Y		N	N	Y
title	Text	Y	Y		**Y***	Y	
status	Picklist	Y	Y		**Y***	Y	
description	Text Area	Y	Y		Y	Y	
measurement	Text Area	Y	Y		Y	Y	
corpGoalLevel	Picklist	Y	Y		Y	Y	
goalYear	Picklist	Y	Y		Y	Y	
creationDate	Date	Y	Y		N	N	
lastUpdated	Date	Y	Y		N	N	

Y* = required for creation.

Relationship URLs:

Relationship	URL
Status	…/object/companygoal/<corpGoalId>/status
History Log	…/object/companygoal/<corpGoalId>/historylog

Sample Code Company Goals

Sample code below references company goals.

Method	Action	Sample Request
GET	Get Company Goal by ID	…/object/companygoal/1
GET	Get all Company Goals	…/object/companygoal
DELETE	Delete Company Goal by ID	…/object/companygoal/1

POST	Create Company Goal	…/object/companygoal
		
		{
		“companygoal”:{
		“description”:”Test Automation Goal Description”,
		“corpGoalLevel”:”Company”,
		“goalYear”:”2011”,
		“measurement”:”Test Automation Goal Measurement”,
		“status”:2,
		“title”:”Test Automation Goal”
		}
		}
		
		OR for XML
		
		<root>
		<companygoal>
		<description>Test Automation Goal Description</description>
		<corpGoalLevel>Company</corpGoalLevel>
		<goalYear>2011</goalYear>
		<measurement>Test Automation Goal Measurement</measurement>
		<status>2</status>
		<title>Test Automation Goal</title>
		</companygoal>
		</root>

POST	Upsert Employee Goal	…/object/companygoal?upsert=true
		
		{
		“companygoal”:{
		“status”:”1”,
		“corpGoalId”:78
		}
		}
		
		OR for XML
		
		<root>
		<companygoal>
		<measurement>updated measurement</measurement>
		<corpGoalId>true</corpGoalId>
		</companygoal>
		<root>

PUT	Update Employee Goal	…/object/employeegoal/78
		
		{
		“companygoal”:{
		“status”:”2”
		}
		}
		
		OR for XML

		<root> <companygoal> <status>100</status> </companygoal> <root>

WEB SERVICE REFERENCE: COMPETENCY LIBRARY

Please Note: Competency Library does not include a search function, as such a Get All execution has been provided.

Only custom competencies are editable within the Taleo Business Edition application, standard competencies cannot be edited.

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Corporate Goals Resource		✔	✔	×	✔	✔	✔
competencyId	Integer	Y	Y		N	N	Y
creationDate	Date	Y	Y		N	N	
lastUpdated	Date	Y	Y		N	N	
name	Text	Y	Y		**Y***	Y	
alternateName	Text	Y	Y		Y	Y	
isSystem	Picklist	Y	Y		N	N	
competencyCategory	Picklist	Y	Y		Y	Y	
description	Text	Y	Y		Y	Y	
createdById	Integer	Y	Y		N	N	
Factor 1	Text Area	Y	Y		**Y***	Y	
Factor 2	Text Area	Y	Y		Y	Y	
Factor 3	Text Area	Y	Y		Y	Y	
Factor 4	Text Area	Y	Y		Y	Y	

Y* = required for creation.

Relationship URLs:

Relationship	URL
Status	…/object/competency/<competencyId>/status

Sample Code Competency Library

Sample code below references competency library items.

Method	Action	Sample Request
GET	Get Company Goal by ID	…/object/competency/1
GET	Get Competency by Name	object/competency?name=Communications
GET	Get all Competencies	…/object/competency
DELETE	Delete Competency by ID	…/object/competency/1 (for custom created competencies only)

POST	Create Competency	…/object/competency
		
		{
		“competency”:{
		“alternateName”:”Managers”,
		“competencyCategory”:”Job Specific”,
		“description”:”Direct Management Experience”,
		“name”:”Management”,
		“factors”:{
		“factor6”:”Assisting”,
		“factor5”:”Presenting”,
		“factor4”:”Communicating”,
		“factor3”:”Mentoring”,
		“factor2”:”Leading”,
		“factor1”:”Managing”
		}
		}
		}
		
		OR for XML
		
		<root>
		<competency>
		<alternateName>Managers</alternateName>
		<competencyCategory>Job Specific</competencyCategory>
		<description>Direct Management Experience</description>
		<name>Management</name>
		<factors>
		<factor6>Assisting</factor6>
		<factor5>Presenting</factor5>
		<factor4>Communicating</factor4>
		<factor3>Mentoring</factor3>
		<factor2>Leading</factor2>
		<factor1>Managing</factor1>
		</factors>
		</competency>
		</root>

POST	Upsert Competency	…/object/competency?upsert=true
		
		{
		“competency”:{
		“name”:”Assistant Manager”,

		“competencyId”:56 } } **OR for XML** <root> <competency> <competencyId>56</competencyId> <alternateName>Managers</alternateName> <competencyCategory>Job Specific</competencyCategory> <description>Direct Management Experience</description> <name>Management</name> </competency> </root>

PUT	Update Competency	…/object/competency/56
		
		{
		“competency”:{
		“name”:”Assistant Manager”
		}
		}
		
		OR for XML
		
		<root>
		<competency>
		<competencyId>56</competencyId>
		<alternateName>Managers</alternateName>
		<competencyCategory>Job Specific</competencyCategory>
		<description>Direct Management Experience</description>
		<name>Management</name>
		</competency>
		</root>

WEB SERVICE REFERENCE: EMPLOYEE ONBOARD PACKETS

Taleo Business Edition Onboard allows for assignment of activities to an employee for onboarding or offboarding with your organization. These activities are bundled into a packet prior to assignment to the employee. The following section discusses the overlying packet of onboarding or offboarding activities through the API, while the next section will outline the actual activities.

Please Note: packet API is for retrieving solely. OnBoarding packet creation through the API is not currently available. For full description of packet API, please retrieve the object information and description details:

…/object/info/**packet**

…/object/**packet**/description/standard

…/object/**packet**/description/custom

To retrieve an onboarding packet by id:

.../object/**packet/{packetId}**

To retrieve the onboarding packet by employee id:

.../object/packet?**employeeId={employeeId}**

To retrieve the JSON hyperschema onboarding packet description deteails:

.../object/**packet**/hyperschema.json

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Packet		✔	×	×	×	×	×
activityPacketId	Integer	Y					
actions	Table Column	N					
activitiesCompleted	Integer	Y					
activitiesCount	Integer	Y					
createdById	User	Y					
creationDate	Date	Y					
currUserDirReports	Table Column	N					
currUserOnlyOwner	Table Column	N					
currUserSubordinates	Table Column	N					
dueDate	Date	Y					
employeeId	Employee	Y					
lastUpdated	Date	Y					
ownerId	User	Y					
title	Text(200)	Y					
usageCxt	Custom	Y					
progress	Custom	N					
status	Picklist	Y					
*** **CUSTOM FIELDS**		Y					

Relationship URLs:

Relationship	URL
Created by	.../object/packet/<activityPacketId>/createdById
Employee	.../object/packet/<activityPacketId>/employeeId
Owner	.../object/packet/<activityPacketId>/ownerId
Status	.../object/packet/<activityPacketId>/status
Activities	.../object/packet/<activityPacketId>/activity
History Log	.../object/packet/<activityPacketId>/historylog

Sample Code for Packets

Method	Action	Sample Request
GET	Get a single onboarding packet by ID	.../object/packet/23
GET	Get onboarding packet(s) by employee record	.../object/packet?employeeId=22

WEB SERVICE REFERENCE: EMPLOYEE ONBOARD ACTIVITIES (WITH FORMS)

Please Note: There is no JSON hyperschema support for Activities.

Taleo Business Edition activities are tasks associated with onboarding or offboarding an employee. These activities can be searched for throught the appropriate GET API, however POST, PUT and DELETE are not supported with onBoard.

You are able to navigate to activities from employee onBoard packets through the relationship URL's. Alternatively, if you know the activity ID, you can navigate directly:

…/object/**activity/{activityId}**

Example response:


```
{
“response”: {
“activity”: {
“id”: 78,
“assignee”: [
34
],
“activityContact”: null,
“activityDesc”: “This Activity for the Employee”,
“dueDate”: “2012-01-11”,
“activityEmployee”: 34,
“item”: “New Employee Activity”,
“status”: 3,
“title”: “New Employee Activity”,
“activityUser”: null,
“relationshipUrls”: {
“assignee”: “…/object/activity/78/assignee”,
“activityEmployee”: “…/object/activity/78/activityEmployee”,
“status”: “…/object/activity/78/status”,
“attachments”: “…/object/activity/78/attachment”,
“formDownloadUrl”: “…/object/activity/78/form/download”
}
}
},
“status”: {
“detail”: {},
“success”: true
}
}
```


If specifically using onBoard activities to retrieve completed 77vailab forms, like i9's or w4's then you will need to first search for your completed employee activity: …/object/activity/search?status=3. The search result will include an activity form download URL in the activity relationship URLs: “formdownload”: “…/object/activity/43/completedform/42/download”. Simply use the formdownload url link to download the completed employee signed form: …/object/activity/43/form/download

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Activities		✔	✔	✔	×	×	×
id	Integer	Y	Y	Y			
actions	Table Column	N	N	N			
ewsAction	Table Column	N	N	N			
activityType	Table Column	N	N	N			

assignee	Text	Y	Y	N
activityContact	Custom	Y	Y	N
currEmployeeUser	Table Column	N	N	N
dependsOn	Table Column	N	N	N
activityDesc	Text	Y	Y	N
dueDate	Date	Y	Y	Y
activityEmployee	Custom	Y	Y	N
fullName	Employee Name	Y	Y	N
item	Text	Y	Y	N
nonCompletedActv	Table Column	N	N	N
progress	Progress Bar Column	N	N	N
restrictions	Table Column	N	N	N
activityPdf	Table Column	N	N	N
status	Picklist	Y	Y	Y
title	Title	Y	Y	Y
activityUser	Custom	Y	Y	Y
***** CUSTOM FIELDS**		Y	Y	Y

Relationship URLs:

Relationship	URL
Assignee	…/object/activity/<id>/assignee
Employee	…/object/activity/<id>/activityEmployee
Status	…/object/activity/<id>/status
Attachments	…/object/activity/<id>/attachment
History Log	…/object/activity/<id>/historylog

Sample Code Onboard Activities

Sample code below references employee activites.

Please Note: By default the API outputs assumes JSON. If XML is requested, please ensure adding the attribute of .xml to the end of the URL's in your request, including in the samples provided below.

Method	Action	Sample Request
GET	Get Activities by ID	…/object/activity/1
GET	Search for Activities	…/object/activity?status=3&dueDate_from=2011-01-11&dueDate_to=2012-01-23

WEB SERVICE REFERENCE: CANDIDATE ADMINISTRATION

Please Note: For Searching of candidates, all Integer, Date and Currency fields should include Parameters <fieldName>_from and/or <fieldName>_to. Please refer to the URL Structure section of this document for full list of parameters and other notes on searching.

Please Note: For export of State code, when both Name and Code are used and seperated by a Pipe (I) symbol, the "Use State Code on REST API Exports of Candidates" checkbox option on the Candidate Picklist Edit: State/Territory page will determine if the two character state code is exported or the state name.

Encrypted fields are only supported with GET if the API User has 'Confidential Data Access' (confidentialAccess) defined as true.

Candidate work history, residence history, references, education and licenses/certification data is retrievable via the Candidate Rolling Entities calls.

cwsPassword – Candidate Website Password field will come back masked in responses for security reasons. You may overwrite a cwsPassword by including it in your PUT/POST call, however it will always be masked following (in GET call).

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Candidates		✔	×	✔	✔	✔	✔
candId	Integer	Y		Y	N	N	Y
candAcceptedOfferPDF	File Upload	N		N	N	N	
candAcceptedOfferId	Offer	Y		N	N	N	
candStartStatus	Picklist	Y		Y	N	N	
cc305PDFCandidate	Custom	N		N	N	N	
creationDate	Date	Y		Y	Y	Y	
referredById	User	Y		N	Y	Y	
candReqAce	Check Box	Y		Y	N	N	
cws	Text	Y		Y	N	N	
city	Text(30)	Y		Y	Y	Y	
country	Picklist	Y		Y	Y	Y	
County	Text(30)	Y		Y	Y	Y	
resumeText	Text Area	Y		N	Y	Y	
birthdate	Encrypted Date	Y		N	Y	Y	
email	Email	Y		Y	Y	Y	
employee	Employee	Y		Y	Y	Y	
firstName	Text(30)	Y		Y	Y	Y	
flagged	Check Box	Y		Y	Y	Y	
gender	Picklist	Y		Y	Y	Y	
googleSearch	Web Link	Y		N	N	N	
hiredDate	Date	Y		Y	Y	Y	
hiredForReqDepartment	Picklist	Y		Y	N	N	
hiredForReqId	Integer	Y		Y	N	N	
hiredForReqJobCode	Text(20)	Y		Y	N	N	
hiredForReqLocation	Picklist	Y		Y	N	N	
hiredForReqTitle	Text(150)	Y		Y	N	N	
inStatusDate	Date	Y		Y	N	N	
IndividualWithDisability	Picklist	Y		Y	Y	Y	
lastName	Text(30)	Y		Y	**Y***	Y	
lastUpdated	Date	Y		Y	N	N	
licenseNumber	Encrypted Text	Y		N	Y	Y	
linkedInSearch	Web Link	Y		N	N	N	
ReasonRej	Picklist	Y		Y	Y	Y	

status	Picklist	Y	Y	Y	Y
rank	Integer	Y	Y	Y	Y
martialStatus	Picklist	Y	Y	Y	Y
middleInitial	Text(30)	Y	Y	Y	Y
nameSuffix	Text(30)	Y	Y	Y	Y
80vailable	Phone	Y	Y	Y	Y
cwsPassword	Password	N	N	Y	Y
passpartNumber	Eencrypted Text	Y	N	Y	Y
phone	Phone	Y	Y	Y	Y
preferredLocale	Picklist	Y	Y	Y	Y
race	Picklist	Y	Y	Y	Y
referredBy	Text(50)	Y	Y	Y	Y
religion	Picklist	Y	Y	Y	Y
Salutation	Text(30)	Y	Y	Y	Y
ssn	Encrypted Text	Y	N	Y	Y
source	Picklist	Y	Y	Y	Y
startDate	Date	Y	Y	Y	Y
state	Picklist	Y	Y	Y	Y
address	Text(100)	Y	Y	Y	Y
address2	Text(100)	Y	Y	Y	Y
veteran	Picklist (multiple)	Y	Y	Y	Y
legalStatus	Picklist	Y	Y	Y	Y
zipCode	Text(10)	Y	Y	Y	Y
ebsEmployeeNumber	Text(100) * If enabled in TBE	Y	Y	Y	Y
ebsIntegrationStatus	Picklist * If enabled in TBE	Y	Y	Y	Y
fusionEmployeeNumber	Text(100) * If enabled in TBE	Y	Y	Y	Y
fusionIntegrationStatus	Picklist * If enabled in TBE	Y	Y	Y	Y
jdeEmployeeNumber	Text(100) * If enabled in TBE	Y	Y	Y	Y
jdeIntegrationStatus	Picklist * If enabled in TBE	Y	Y	Y	Y
nsuiteEmployeeNumber	Text(100) * If enabled in TBE	Y	Y	Y	Y
nsuiteIntegrationStatus	Picklist * If enabled in TBE	Y	Y	Y	Y
psoftEmployeeNumber	Text(100) * If enabled in TBE	Y	Y	Y	Y
psoftIntegrationStatus	Picklist * If enabled in TBE	Y	Y	Y	Y
resumeFileName	Text	Y	N	N	N
resumeContentType	Text	Y	N	N	N
addedWithin	Picklist	N	Y	N	N
updatedWithin	Picklist	N	Y	N	N
***** CUSTOM FIELDS**		Y	Y	Y	Y

Y* = required for creation.

Relationship URLs:

Relationship	URL
Referred By ID	…/object/candidate/<candId>/referredById
Employee	…/object/candidate/<candId>/employee
Status	…/object/candidate/<candId>/status

Requisition	…/object/candidate/<candId>/requisition
Attachments	…/object/candidate/<candId>/attachment
Resume	…/object/candidate/<candId>/resume
Interview*	…/object/candidate/<candId>/interview
Work History	…/object/candidate/<candId>/workhistory
References	…/object/candidate/<candId>/reference
Education	…/object/candidate/<candId>/education
Residence	…/object/candidate/<candId>/residence
Certificates	…/object/candidate/<candId>/certificate
Background Checks	…/object/candidate/<candId>/backgroundcheck
Offer	…/object/candidate/<candId>/offer
Expense	…/object/candidate/<candId>/expense
Comment*	…/object/candidate/<candId>/comment
History Log*	…/object/candidate/<candId>/historylog
Contact Log*	…/object/candidate/<candId>/contactlog

*Not supported in the JSON Hyperschema.

Sample Code Candidate Record

Sample code below references candidate record.

Please Note: By default the API outputs and assumes JSON. If XML is requested, please ensure adding the attribute of .xml to the end of the URL's in your request, including in the samples provided below.

For upsert parameter used to conduct a record update, even though you do not need to state the ID in the request URL, you will need to provide the candidate ID in the message body for update to match an existing record.

For update you can only send back the specific fields you are interested in updating.

Method	Action	Sample Request
GET	Get Candidate by ID	…/object/candidate/1
GET	Search for Candidate	…/object/candidate/search?start=1&limit=1&addedWithin=5000&status=1&fields=status|firstName
DELETE	Delete Candidate by ID	…/object/candidate/1

POST	Create Candidate	…/object/candidate
		
		{
		"candidate": {
		"city": "San Francisco",
		"country": "US",
		"resumeText": "test test test",
		"email": "tqatest9@invalidemail.com",
		"firstName": "Test9",
		"lastName": "Qatest9",
		"status": 2,
		"middleInitial": "F",
		"81vailable": "415-256-5219",
		"race": "White (not Hispanic or Latino)",
		"source": "Careers Website",
		"state": "US-CA",
		"address": "653 New Ave",
		"veteran": [

		“None”], “zipCode”: “94122” } } **OR for XML** <root> <candidate> <city>San Francisco</city> <source>Careers Website</source> <lastName>Careers Website</lastName> <firstName>Careers Website</firstName> <email>test@invalidemail.com</email> <status>New</status> <state>US-CA</state> <address>653 New Ave</address> </candidate> </root>

POST	Upsert Candidate	…/object/candidate?upsert=true
		
		{
		“candidate”:{
		“city”: “San Francisco”,
		“country”: “US”,
		“resumeText”: “test test test”,
		“email”: “tqatest9@invalidemail.com”,
		“firstName”: “Test9”,
		“candId”: “50”,
		“lastName”: “Qatest9”,
		“status”: 2,
		“middleInitial”: “F”,
		“82vailable”: “415-256-5219”,
		“race”: “White (not Hispanic or Latino)”,
		“source”: “Careers Website”,
		“state”: “US-CA”,
		“address”: “653 New Ave”,
		“veteran”: [
		“None”
		],
		“zipCode”: “94122”
		}
		}
		
		OR for XML
		
		<root>
		<candidate>

		<lastName>Sanchez</lastName> < candId>15</candId> <status>New</status> <hourlyWage>19.00</hourlyWage> <employee> </candidate> </root>

PUT	Update Candidate	…/object/candidate/15
		
		{
		“candidate”:{
		“city”: “San Francisco”,
		“country”: “US”,
		“resumeText”: “test test test”,
		“email”: “tqatest9@invalidemail.com”,
		“firstName”: “Test9”,
		“candId”: “50”,
		“lastName”: “Qatest9”,
		“status”: 2,
		“middleInitial”: “F”,
		“83vailable”: “415-256-5219”,
		“race”: “White (not Hispanic or Latino)”,
		“source”: “Careers Website”,
		“state”: “US-CA”,
		“address”: “653 New Ave”,
		“veteran”: [
		“None”
		],
		“zipCode”: “94122”
		}
		}
		
		OR for XML
		
		<root>
		<candidate>
		<lastName>Sanchez</lastName>
		< candId>15</candId>
		<status>New</status>
		<hourlyWage>19.00</hourlyWage>
		<employee>
		</candidate>
		</root>

WEB SERVICE REFERENCE: CANDIDATE ROLLING ENTITIES

The Taleo Business Edition candidate rolling entities are dynamic forms of historic talent information that is tied to a single candidate. These are:

- Work History

- Education
- Licenses & Certifications
- References
- Previous Addresses

There is no search associated with rolling entities, however a GET ALL function that is tied to a specific candidate has been provided which is specific to a candidate record. To access them, the URL format would be:

<<HOST_URL>>/object/candidate/**<<CANDIDATE_ID>>**/**<<ROLLING_ENTITY_NAME>>**

Where the rolling entity name would be:

- workhistory
- education
- certificate
- reference
- residence

If a single instance is requested, use the rolling entity ID post the URL:

<<HOST_URL>>/object/candidate/**<<CANDIDATE_ID>>**/**<<ROLLING_ENTITY_NAME>>/<<ROLLING_ENTITY_ID>>**

The getMetadata call for a rolling entity would be for standard fields only vs including custom fields:

<<HOST_URL>>/object/candidate/**<<ROLLING_ENTITY_NAME>>**/description/**standard**

<<HOST_URL>>/ object/candidate/**<<ROLLING_ENTITY_NAME>>**/description/**custom**

Work History Actions

Please note the following supported RESTful actions:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Work History Resource		✔	✔	×	✔	✔	✔
id	Integer	Y	Y		Y	Y	Y (Delete by ID)
workHistCityState	Text(100)	Y	Y		Y	Y	
workHistStreet	Text(100)	Y	Y		Y	Y	
workHistCoName	Text(100)	Y	Y		Y	Y	
workHistPhone	Phone	Y	Y		Y	Y	
workHistDesc	Text (2500)	Y	Y		Y	Y	
workHistSupervisor	Text(100)	Y	Y		Y	Y	
workHistExplanation	Text(100)	Y	Y		Y	Y	
workHistPay	Text(20)	Y	Y		Y	Y	
workHistFrom	Text[1]	Y	Y		Y	Y	
workHistOkContact	Check Box	Y	Y		Y	Y	
workHistReasonForLeaving	Picklist	Y	Y		Y	Y	
workHistSuperTitle	Text(100)	Y	Y		Y	Y	
workHistJobTitle	Text(100)	Y	Y		Y	Y	
workHistTo	Text[1]	Y	Y		Y	Y	
***** CUSTOM FIELDS**		Y	Y		Y	Y	

Notes:

1. **workHistFrom and workHistTo expect a format of 'MM-YYYY', for example '09-2013', or the value 'To Present'. Values provided are validated.**

Education Actions

Please note the following supported RESTful actions:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Education Resource		✔	✔	×	✔	✔	✔
id	Integer	Y	Y		Y	Y	Y (Delete by ID)
eduHistCity	Text	Y	Y		Y	Y	
eduHistFrom	Text[1]	Y	Y		Y	Y	
eduHistTo	Text[1]	Y	Y		Y	Y	
eduHistDegreeAchieved	Text	Y	Y		Y	Y	
eduHistFieldOfStudy	Text	Y	Y		Y	Y	
eduHistSchool	Text	Y	Y		Y	Y	
eduHistState	Picklist	Y	Y		Y	Y	
***** CUSTOM FIELDS**		Y	Y		Y	Y	

Notes:

1. **eduHistFrom and eduHistTo expect a format of 'MM-YYYY', for example '09-2013', or the value 'To Present'. Values provided are validated.**

Licenses & Certificate Actions

Please note the following supported RESTful actions:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Licenses Resource		✔	✔	×	✔	✔	✔
id	Integer	Y	Y		Y	Y	Y (Delete by ID)
certName	Text	Y	Y		Y	Y	Y
certIssuingBody	Text	Y	Y		Y	Y	Y
certYear	Text(4)	Y	Y		Y	Y	Y
***** CUSTOM FIELDS**		Y	Y		Y	Y	Y

References Actions

Please note the following supported RESTful actions:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
References Resource		✔	✔	×	✔	✔	✔
id	Integer	Y	Y		Y	Y	Y (Delete by ID)
refEmail	Email	Y	Y		Y	Y	
refPhone	Phone	Y	Y		Y	Y	
refName	Text	Y	Y		Y	Y	
refYearsKnown	Text(2)	Y	Y		Y	Y	
***** CUSTOM FIELDS**		Y	Y		Y	Y	

Previous Addresses Actions

Please note the following supported RESTful actions:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Address Resource		✔	✔	×	✔	✔	✔
id	Integer	Y	Y		Y	Y	Y (Delete by ID)
resHistCity	Text	Y	Y		Y	Y	
resHistCountry	Picklist	Y	Y		Y	Y	
resHistFrom	Text[1]	Y	Y		Y	Y	
resHistState	Picklist	Y	Y		Y	Y	
resHistStreet	Text	Y	Y		Y	Y	
resHistTo	Text[1]	Y	Y		Y	Y	
resHistZipCode	Text	Y	Y		Y	Y	
***** CUSTOM FIELDS**		Y	Y		Y	Y	

Notes:

1. **resHistFrom and resHistTo expect a format of 'MM-YYYY', for example '09-2013', or the value 'To Present'. Values provided are validated.**

Sample Code Candidate Rolling Entities

Sample code below references a work history instance; however the URL structure and available actions are the same for the other candidate rolling entities.

Please Note: By default the API outputs and assumes JSON. If XML is requested, please ensure adding the attribute of .xml to the end of the URL's in your code, including in the samples provided below.

Method	Action	Sample Request
GET	Get single Candidate work history by ID	…/object/candidate/41/workhistory/2112
GET	Get all work history for a single candidate	…/object/candidate/41/workhistory/
DELETE	Delete single work history instance	…/object/candidate/41/workhistory/2112

POST	Create work history for an candidate	…/object/candidate/41/workhistory
		
		{
		“workhistory”:
		{
		“workHistCityState”: “US-CA”,
		“workHistCoName”: “Taleo”,
		“workHistPhone”: “925-483-2342”,
		“workHistStreet”: “10 Happy Street”,
		“workHistFrom”: “09-1997”,
		“workHistTo”: “To Present”,
		“workHistDesc”: “Best Manager”,
		“workHistSupervisor”: “Jason”,
		“workHistExplanation”: “Perfect Manager”,
		“workHistPay”: “14.00”,
		“workHistOkContact”: true,
		“workHistSuperTitle”: “Manager”,
		“workHistJobTitle”: “Manager”
		}
		}
		}
		
		OR for XML
		
		<root>
		<workhistory>
		<workHistCityState>US-CA</workHistCityState>
		<workHistCoName>Taleo</workHistCoName>
		<workHistPhone>925-111-1111</workHistPhone>
		<workHistStreet>945 Dublin Blvd</workHistStreet>
		<workHistFrom>09-1997</workHistFrom>
		<workHistTo>To Present</workHistTo>
		<workHistDesc>Wrestler</workHistDesc>
		<workHistSupervisor></workHistSupervisor>
		<workHistExplanation></workHistExplanation>
		<workHistPay></workHistPay>

		<workHistOkContact>false</workHistOkContact> <workHistReasonForLeaving></workHistReasonForLeaving> <workHistSuperTitle></workHistSuperTitle> <workHistJobTitle>sDBV</workHistJobTitle> </workhistory> </root>

POST	Upsert work history for an candidate	…/object/candidate/41/workhistory?upsert=true
		
		{
		“workhistory”:
		{
		“workHistCityState”: “US-CA”,
		“workHistCoName”: “Taleo”,
		“workHistPhone”: “925-483-2342”,
		“workHistStreet”: “10 Happy Street”,
		“workHistFrom”: “09-1997”,
		“workHistTo”: “To Present”,
		“workHistDesc”: “Best Manager”,
		“workHistSupervisor”: “Jason”,
		“workHistExplanation”: “Perfect Manager”,
		“workHistPay”: “14.00”,
		“workHistOkContact”: true,
		“workHistSuperTitle”: “Manager”,
		“workHistJobTitle”: “Manager”,
		“id”: 2112
		}
		}
		}
		
		OR for XML
		
		<root>
		<workhistory>
		<workHistCityState></workHistCityState>
		<workHistCoName></workHistCoName>
		<workHistPhone></workHistPhone>
		<workHistStreet></workHistStreet>
		<workHistFrom>09-1997</workHistFrom>
		<workHistTo>To Present</workHistTo>
		<workHistDesc></workHistDesc>
		<workHistSupervisor></workHistSupervisor>
		<workHistExplanation></workHistExplanation>
		<workHistPay></workHistPay>
		<workHistOkContact>false</workHistOkContact>
		<workHistReasonForLeaving></workHistReasonForLeaving>
		<workHistSuperTitle></workHistSuperTitle>
		<workHistJobTitle>sDBV</workHistJobTitle>
		<id>2112</id>
		</workhistory>
		</root>

PUT	Update a field for a work history for a candidate	…/object/candidate/41/workhistory/2112
		
		{
		“workhistory”:
		{
		“workHistCityState”: “US-CA”,
		“workHistCoName”: “Taleo”,
		“workHistPhone”: “925-483-2342”,
		“workHistStreet”: “10 Happy Street”,
		“workHistFrom”: “09-1997”,
		“workHistTo”: “To Present”,
		“workHistDesc”: “Best Manager”,
		“workHistSupervisor”: “Jason”,
		“workHistExplanation”: “Perfect Manager”,
		“workHistPay”: “14.00”,
		“workHistOkContact”: true,
		“workHistSuperTitle”: “Manager”,
		“workHistJobTitle”: “Manager”,
		“id”: 2112
		}
		}
		}
		
		OR for XML
		
		<root>
		<workhistory>
		<workHistTo>To Present</workHistTo>
		<workHistFrom>09-1997</workHistFrom>
		<id>2112</id>
		</workhistory>
		</root>

WEB SERVICE REFERENCE: CANDIDATE APPLICATIONS

Candidate application API specifically references Taleo Business Edition’s cand-req table. This relationship table sits in the middle of the candidate table and requisition table to outline which positions a candidate is tied to and what their workflow status is for that requisition.

NEW is 13.3: Candidate Applications now support Custom Fields that can be configured in the application.

<<HOST_URL>>/object/candidateapplication/description/standard

<<HOST_URL>>/object/candidateapplication/description/custom

As a relationship table between candidates and requisitions, there are 2 parameters that can be sent for usage of GET ALL specific to candidate or requisition:

<<HOST_URL>>/object/candidateapplication?**candidateId=XX**

<<HOST_URL>>/object/candidateapplication?**requisitionId=XX**

Additionally, if requesting a specific GET for a candidate to a requisition instance, then either use the ID or pass both parameters:

<<HOST_URL>>/object/candidateapplication?**candidateId=XX&requisitionId=XX**

<<HOST_URL>>/object/candidateapplication/10

For PUT or POST of candidate application, a parameter can be sent called 'doRanking'. This Boolean value will initiate a rank evaluation when the candidate has been newly appended to a requisition. The rank score will be calculated and displayed within the candidateapplication (cand-req table). For example, if the requisition has an automated rank for keywords a candidate has in their profile, this doRanking attribute will mimic this calc for new associations via the API.

<<HOST_URL>>/object/candidateapplication?**doRanking=true**

Candidate Application Actions

Please note the following supported RESTful actions:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Candidate Application Resource		✔	✔	×	✔	✔	✔
id	Integer	Y	Y		Y	Y	Y (Delete by ID)
candidateId	Integer	Y	Y		**Y***	Y	
requisitionId	Integer	Y	Y		**Y***	Y	
candReqInStatusDate	Date	Y	Y		N	N	
preRejectionStatus	Picklist	Y	Y		N	N	
status	Picklist	Y	Y		Y	Y	
reasonRejected	Picklist	Y	Y		Y	Y	
rank	Integer	Y	Y		Y	Y	
isACE	Check Box	Y	Y		N	N	
dateApplied	Date	Y	Y		Y	Y	
hireDate	Date	Y	Y		Y	Y	
startDate	Date	Y	Y		Y	Y	
referredBy	Text	Y	Y		Y	Y	
Source	Picklist	Y	Y		Y	Y	
careersWebSiteId	Text	Y	Y		N	N	
eSignature	Text	Y	Y		N	N	
eSignatureDate	Date	Y	Y		N	N	
resumeFileName	String	Y	Y		N	N	
resumeContentType	String	Y	Y		N	N	
coverLetter	Text Area	Y	Y		N	N	
indivWithDisability	Long	Y	Y		N	N	
doRanking	Check Box	N	N		Y	Y	
***** CUSTOM FIELDS**		Y	Y		Y	Y	

Y* = required for creation.

Relationship URLs:

Relationship	URL
Status	…/object/candidateapplication/<Id>/status
Candidate	…/object/candidate/<candidateId>
Requisition	…/object/requisition/<requisitionId>
Next Steps	…/object/candidateapplication/<Id>/nextsteps
Resume	…/object/candidateapplication/<Id>/resume

Application Snapshot	…/object/candidateapplication/<Id>/candReqApplication
Candidate Answer	…/object/candidateanswer/<candTestId>

Sample Code Candidate Application

Sample code below references candidate applications.

Method	Action	Sample Request
GET	Get single Candidate Application Reference by ID	…/object/candidateapplication/3
GET	Get all applications for a specific candidate	…/object/candidateapplication?candidateId=54
GET	Get all candidates that have applied for a specific requisition.	…/object/candidateapplication?requisitionId=43
DELETE	Delete single candidate application instance	…/object/candidateapplication/3

POST	Create work history for a candidate	…/object/candidateapplication
		
		{
		“candidateapplication”: {
		“candidateId”: 41,
		“requisitionId”: 43,
		“status”: 2,
		“reasonRejected”: -1,
		“rank”: 100,
		“dateApplied”: “2012-04-30”
		}
		}
		
		OR for XML
		
		<root>
		<candidateapplication>
		<candidateId>41</candidateId>
		<requisitionId>69</requisitionId>
		<status>2</status>
		<reasonRejected>-1</reasonRejected>
		<rank>10</rank>
		<dateApplied>2012-04-20</dateApplied>
		</candidateapplication>
		</root>

POST	Upsert candidate application for an candidate	…/object/candidateapplication?upsert=true
		
		{
		“candidateapplication”: {
		“candidateId”: 41,
		“requisitionId”: 43,
		“status”: 2,
		“reasonRejected”: -1,
		“rank”: 100,
		“dateApplied”: “2012-04-30”
		}

		} **OR for XML** <root> <candidateapplication> <candidateId>41</candidateId> <requisitionId>69</requisitionId> <status>2</status> <reasonRejected>-1</reasonRejected> <rank>10</rank> <dateApplied>2012-04-20</dateApplied> </candidateapplication> </root>

PUT	Update a candidate application	…/object/candidateapplication/1281
		
		{
		“candidateapplication”: {
		“candidateId”: 41,
		“requisitionId”: 43,
		“status”: 2
		}
		}
		
		OR for XML
		
		<root>
		<candidateapplication>
		<candidateId>41</candidateId>
		<requisitionId>69</requisitionId>
		<status>2</status>
		<reasonRejected>-1</reasonRejected>
		<rank>10</rank>
		<dateApplied>2012-04-20</dateApplied>
		<id>1281</id>
		</candidateapplication>
		</root>

WEB SERVICE REFERENCE: CANDIDATE ANSWER TO SCREENING QUESTIONS

The Taleo Business Edition Candidate Answer entity is used for storing and managing candidate answers to screening questions, attached to a requisition.

The field definitions for Candidate Answer are available through the following discovery calls (standard fields only through the first URL vs including custom fields through the second provided URL):

<<HOST_URL>>/object/**candidateanswer**/description/standard

<<HOST_URL>>/object/**candidateanswer**/description/custom

For JSON hyperschema field definition, use the following URL:

<<HOST_URL>>/object/**candidateanswer**/hyperschema.json

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Candidate Answer		✔	✔	×	✔	×	×
candTestId	Integer	Y	Y				
candidateId	Integer	Y	Y		Y*		
requisitionId	Integer	Y	Y		Y*		
questionId	Integer	Y	Y		Y*		
answer	String (Boolean, Text, Picklist)	Y	Y		Y*		

Y* = required for creation

Relationship	URL
Candidate Application	…/object/candidateapplication?**candidateId={candId}&requisitionId={redId}**

For retrieving candidate answer of all screening questions for a requisition, use the following URLs:

<<HOST_URL>>/object/candidateanswer/{candTestId}

For retrieving candidate answer of a particular screening question for a requisition, use the following URLs:

<<HOST_URL>>/object/candidateanswer/{candTestId}?questionId={questionId}

Example response:


```
        {
"response": {
    "candidateanswers": [
        {
            "candidateanswer": {
                "candTestId": 111,
                "questionId": 48,
                "answer": Los Angeles,
                "relationshipUrls": {
                    "collection": "https://qa.tbe.taleocloud.net/qa1/ats/api/v1/object/candidateanswer/111",
                    "instance": "https://qa.tbe.taleocloud.net/qa1/ats/api/v1/object/candidateanswer/111?questionId=48"
                }
            }
        }
    ]
},
"status": {
    "success": true,
```

```
        "detail": {}
      }
    }
```


Sample Code Approvals

Method	Action	Sample Request
GET	Get candidate answer of a particular screening question for a requisition	.../object/candidateanswer/111?questionId=48
GET	Get candidate answer of all screening questions for a requisition	.../object/candidateanswer/111

POST	Submit/Create Candidate Answer for Requisition screening question	../object/candidateanswer
		
		{
		"candidateanswer" : {
		"candidateId" : "92",
		"requisitionId" : "66" ,
		"questionId":"48",
		"answer" : "Los Angeles "
		}
		}
		
		or For XML
		
		<root>
		<candidateanswer>
		<candidateId>92</candidateId>
		<requisitionId>66</requisitionId>
		<questionId>-48</questionId>
		<answer>Los Angeles </answer>
		</candidateanswer>
		</root>

POST	Upsert an Approval	../object/candidateanswer?upsert=true
		
		{
		"candidateanswer" : {
		"candidateId" : "92",
		"requisitionId" : "66" ,
		"questionId":"48",
		"answer" : "Los Angeles"
		}
		}
		
		or For XML
		
		<root>
		<candidateanswer>
		<candidateId>92</candidateId>
		<requisitionId>66</requisitionId>
		<questionId>-48</questionId>

		<answer>Los Angeles </answer> </candidateanswer> </root>

WEB SERVICE REFERENCE: CANDIDATE OFFERS

The Taleo Business Edition candidate offers for storing information on Offers for Candidates against Requisitions.

Note: Search is not currently supported for Offers. However, the collection of offers for a Candidate can be retrieved by specifying the Candidate ID as a parameter to the GET method.

Example: **<<HOST_URL>>**/object/offer.xml?candidateId={candidateId}

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Offers		✔	×	×	✔	×	✔
offerId	Integer	Y			N		Y (Delete record by ID)
apprStatus	Custom	Y			N		
candidate	Custom	Y			**Y***		
currUserApprStatus	Check Box	Y			N		
duration	Picklist	Y			**Y***		
expirDate	Date	Y			Y		
inStatusDate	Date	Y			N		
manager	Picklist	Y			**Y***		
template	Picklist	Y			N		
payRate	Text (50)	Y			Y		
requisition	Picklist	Y			**Y***		
startDate	Date	Y			Y		
status	Picklist	Y			Y		
stockOptions	Text (50)	Y			Y		
title	Text (100)	Y			**Y***		
offerFile	File Upload	N			N		

Y* = required for creation.

Relationship URLs:

Relationship	URL
Candidate	.../object/offer/<offerId>/candidate

Manager	…/object/offer/<offerId>/manager
Offer Letter	…/object/offer/<offerId>/offer
Requisition	…/object/offer/<offerId>/requisition
Status	…/object/offer/<offerId>/status
Approval	…/object/offer/<offerId>/approval

Sample Code Candidate Offers

Sample code below references candidate offers.

Method	Action	Sample Request
GET	Retrieves metadata	…/object/info/offer
GET	Retrieves standard fields	…/object/offer/description/standard
GET	Retrieves custom fields	…/object/offer/description/custom
GET	Retrieves display field info	…/object/displayfield/OFFER/{96vailable}
GET	Retrieves offers by candidate ID	…/object/offer?candidateId={id}
GET	Retrieves offers by id	…/object/offer/{id}
GET	Retrieves candidate related to offer	…/object/offer/{id}/candidate
GET	Retrieves requisition related to offer	…/object/offer/{id}/requisition
GET	Retrieves offer status	…/object/offer/{id}/status
GET	Retreives offer letter	…/object/offer/{id}/offer
GET	Retrieves offer manager	…/object/offer/{id}/manager
GET	Retrieves all offers related to candidate	…/object/candidate/{id}/offer
GET	Retrieves offer related to candidate by ID	…/object/candidate/{id}/offer/{offerId}
DELETE	Deletes offer	…/object/offer/{id}
POST	Creates offer for candidate	…/object/candidate/{id}/offer
POST	Creates offer letter	…/object/offer/{id}/offer
POST	Upsert Create	../object/offer?upsert=true

POST	Upsert Update	../object/offer?upsert=true
		
		{
		“offer”: {
		“title”: “REST Offer”,
		“candidate”:{candidateId},
		“offerId”: {Id},
		“duration”: “FULLTIME”,
		“manager”: “{userId}”,
		“requisition”: “{requisitionId}” // this must be a requisition that the candidate has applied to
		}
		}

	Sample Offer JSON	{
		“offer”: {
		“title”: “REST Offer”,
		“duration”: “FULLTIME”,
		“manager”: “{userId}”,
		“requisition”: “{requisitionId}” // this must be a requisition that the candidate has applied to
		}
		}

WEB SERVICE REFERENCE: OFFER APPROVALS MANAGEMENT

Some entities in Taleo Business Edition allow for recording approvals or rejections in approvals table. This table is available through the API as well for programmatic access. The Approval resource currently covers approvals for Requisitions and Offers.

For retrieving details of a particular approval, use the following URLs:

<<HOST_URL>>/object/approval/{id}

For retrieving all approvals within an associated entity record, use the following URLs, these are available **as relationship URLs for the entity**:

<<HOST_URL>>/object/**requisition**/{id}/approval

<<HOST_URL>>/object/**offer**/{id}/approval

For retrieving all approval object details, use the following URL

<<HOST_URL>>/object/info

<<HOST_URL>>/object/info/approval

For field definition, use the following URL (please note that custom and standard URL will output the same fields as this is an enclosed table):

<<HOST_URL>>/object/**approval**/description/standard

<<HOST_URL>>/object/**approval**/description/custom

For JSON hyperschema field definition, use the following URL:

<<HOST_URL>>/object/**approval**/hyperschema.json

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Comments		✔	×	×	✔	✔	×
id	Integer	Y			N	N	
creationDate	Date	Y			N	N	
lastUpdated	Date	Y			N	N	
apprStatus	Integer	Y			**Y***	Y	
inStatusDate	Date	Y			N	Y	
entityType	Entity Code	Y			**Y***	Y	
entityId	Entity Id	Y			**Y***	Y	
approver	Integer (userid)	Y			**Y***	Y	
overrideApprover	Integer (userid)	Y			Y	Y	
Text	Text Area	Y			Y	Y	

Y* = required

Relationship URLs:

Relationship	URL
Requisition	…/object/approval/<id>/requisition
Offer	…/object/approval/<id>/offer

approver	…/object/approval/<id>/approver
overrideApprover	…/object/approval/<id>/overrideApprover
status	…/object/approval/<id>/status

Sample Code Approvals

Method	Action	Sample Request
GET	Get Single Approval By ID	…/object/approval/132
GET	Get Approvals for a Requisition (Requisition ID 55)	…/object/**requisition**/55/approval

POST	Create Approval	…/object/approval
		
		{
		“approval”:{
		“ entityType”:”REQU”,
		“entityId”:74,
		“approver”:42,
		“apprStatus”:1,
		“text”:”My approver comments”
		}
		}
		
		OR XML
		
		<root>
		<approval>
		<entityType>REQU</entityType>
		<entityId>74</entityId>
		<apprStatus>1</apprStatus>
		<approver>42</approver>
		<text> My approver comments </text>
		</approval>
		</root>

POST	Upsert an Approval	…/object/approval?upsert=true
		
		{
		“approval”:{
		“id”:”123”,
		“ entityType”:”REQU”,
		“entityId”:74,
		“approver”:42,
		“apprStatus”:1,
		“text”:”My approver comments”
		}
		}
		
		OR XML
		
		<root>
		<approval>

		<id>123</id> <entityType>REQU</entityType> <entityId>74</entityId> <apprStatus>1</apprStatus> <approver>42</approver> <text> My approver comments </text> </approval> </root>
PUT	Update Approval	…/object/approval/123 {“approval”:{“text”:”New approval comment”}} **OR for XML** <root> <approval> <text>New approval comment</text> </approval> </root>

WEB SERVICE REFERENCE: REQUISITION (&TEMPLATE) ADMINISTRATION

The Taleo Business Edition requisitions and requisition templates are used for storing jobs for recruiting. Although requisition and requisition templates are visually separated within the application, from a table or API perspective they are the same with the difference of a Boolean / checkbox field of isTemplate. Where isTemplate=true, the position will be loaded in the requisition template library. Where false or not provided at all, it will be loaded in the standard requisition library. The isTemplate attribute can be used for searching as well, with not provided being the same as false.

To search for a requisition, the URL format would be:

<<HOST_URL>>/object/requisition/search?

When searching through requisitions, all Integer, Date and Currency fields should include Parameters < 99ieldname >_from and/or < 99ieldname>_to. Please refer to the URL Structure section of this document for full list of parameters and other notes on searching.

The field definitions for requisitions are available through the following discovery calls (standard fields only through the first URL vs including custom fields through the second provided URL):

<<HOST_URL>>/object/requisition/description/**standard**

<<HOST_URL>>/ object/ requisition/description/**custom**

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Requisitions		✔	×	✔	✔	✔	✔
reqId	Integer	Y		Y	N	N	Y (Delete record by ID)
actions	Table Column	N		N	N	N	
addedWithin	Picklist	N		Y	N	N	
approvers	Custom	Y		Y	Y	Y	
Budgeted	Picklist	Y		Y	Y	Y	
city	Text	Y		Y	Y	Y	
creationDate	Date	N		Y	N	N	

currUserApprStatus	Check Box	N	N	N	N
currUserDept	Table Column	N	N	N	N
currUserDivision	Table Column	N	N	N	N
currUserLocation	Table Column	N	N	N	N
currUserOnlyOwner	Table Column	N	N	N	N
currUserOwner	Table Column	N	N	N	N
currUserRegion	Table Column	N	N	N	N
currUserRegionLoc	Table Column	N	N	N	N
cws	Picklist	N	Y	N	N
department	Picklist	Y	Y	Y	Y
description	Text Area	Y	Y	Y	Y
division	Picklist (multiple)	Y	Y	Y	Y
duration	Picklist	Y	Y	Y	Y
eQuest Posting Site	Picklist	Y	Y	Y	Y
filledDate	Date	Y	Y	Y	Y
inStatusDate	Date	Y	Y	N	N
isHotJob	Check Box	Y	Y	Y	N
isTemplate	Check Box	Y	Y	Y	Y
jobBrief	Text(255)	Y	N	Y	Y
Job Reason	Picklist	Y	Y	Y	Y
jobCat	Picklist	Y	Y	Y	Y
jobCode	Text(20)	Y	Y	Y	Y
jobFunction	Picklist	Y	Y	Y	Y
jobIndustry	Picklist	Y	Y	Y	Y
jobType	Picklist	Y	Y	Y	Y
keywords	Text	N	N	N	N
lastUpdated	Date	Y	Y	N	N
location	Picklist	Y	Y	**Y***	Y
numCands	Integer	N	Y	N	N
numOpen	Integer	Y	Y	Y	Y
offerApprs	Custom	Y	Y	Y	Y
openedDate	Date	Y	Y	Y	Y
owners	Custom	Y	Y	**Y***	Y
payRange	Text(100)	Y	Y	Y	Y
region	Picklist (multiple)	Y	Y	Y	Y
relevance	Table Column	N	N	N	N
Replacement for	Text	Y	Y	Y	Y
reqCandAce	Check Box	N	Y	N	N
reqCandDateApplied	Date	N	Y	N	N
reqCandidatesIcon	Requisition Candidates Icon	N	N	N	N
reqCandNextSteps	Table Column	N	N	N	N
reqCandRank	Integer	N	Y	N	N
reqCandResRej	Picklist (multiple)	N	Y	N	N
reqCandStatus	Picklist	N	Y	N	N
reqCandStatusAction	Table Column	N	N	N	N
ReqHiringManager	Custom	Y	Y	Y	Y
ReqRecruiter	Custom	Y	Y	Y	Y
state	Picklist	Y	Y	Y	Y
status	Picklist	Y	Y	**Y***	Y
title	Text(150)	Y	Y	**Y***	Y

txReqStatus	Partner field	N	N	N	N
updatedWithin	Picklist	N	Y	N	N
viewers	Custom	N	N	N	N
workflowDefId	Picklist	Y	Y	Y	Y
zipCode	Text	Y	Y	Y	Y
ebsRequisitionNumber	Text(100) * If enabled in TBE	Y	Y	Y	Y
ebsIntegrationStatus	Picklist * If enabled in TBE	Y	Y	Y	Y
fusionRequisitionNumber	Text(100) * If enabled in TBE	Y	Y	Y	Y
fusionIntegrationStatus	Picklist * If enabled in TBE	Y	Y	Y	Y
fusionBusinessUnitId	Picklist * If enabled in TBE	Y	Y	Y	Y
fusionLegalEntityId	Picklist * If enabled in TBE	Y	Y	Y	Y
jdeRequisitionNumber	Text(100) * If enabled in TBE	Y	Y	Y	Y
jdeIntegrationStatus	Picklist * If enabled in TBE	Y	Y	Y	Y
psoftRequisitionNumber	Text(100) * If enabled in TBE	Y	Y	Y	Y
psoftIntegrationStatus	Picklist * If enabled in TBE	Y	Y	Y	Y
reqSetHiredDate	Check Box	Y	N	Y	Y
reqDecrementOpenings	Check Box	Y	N	Y	Y
reqEmailNonHired	Check Box	Y	N	Y	Y
reqChangeStatusNonHired	Check Box	Y	N	Y	Y
reqChangeStatusPosting	Check Box	Y	N	Y	Y
***** CUSTOM FIELDS**		Y	Y	Y	Y

Y* = required for creation.

Relationship URLs:

Relationship	URL
Department	…/object/requisition/<reqId>/department
Location	…/object/requisition/<reqId>/location
Offer Approvers	…/object/requisition/<reqId>/offerApprs
Regions	…/object/requisition/<reqId>/region
Req Approvers	…/object/requisition/<reqId>/approvers
Owners	…/object/requisition/<reqId>/owners
Status	…/object/requisition/<reqId>/status
Attachments	…/object/requisition/<reqId>/attachment
Candidates	…/object/requisition/<reqId>/candidate
Expense	…/object/requisition/<reqId>/expense
Comment	…/object/requisition/<reqId>/comment
Question	…/object/requisition/<reqId>/question
Approval	…/object/requisition/<reqId>/approval
History Log	…/object/requisition/<reqId>/historylog
Contact Log	…/object/requisition/<reqId>/contactlog
Poster	…/object/requisition/<reqId>/poster

Sample Code Requisitions

Method	Action	Sample Request
GET	Get Requisition by ID	…/object/requisition/69

GET	Search for Requisition	…/object/requisition/search?openedDate_from=2008-12-02&duration=FULLTIME&cws=1
DELETE	Delete Requisition by ID	…/object/requisition/69

POST	Create Requisition	…/object/requisition
		
		{
		“requisition”: {
		“title”: “Head Nurse”,
		“status”: “2”,
		“numOpen”: “2”,
		“candReqApplication”: “”,
		“workflowDefId”: 13,
		“department”: 3,
		“description”: “Head Nurse”,
		“jobCat”: “(1.2) First/Mid Level Officials & Managers”,
		“duration”: “FULLTIME”,
		“isTemplate”: true,
		“jobCode”: “NURSE-1293”,
		“location”: 1,
		“offerApprs”: [],
		“openedDate”: “2008-12-02”,
		“payRange”: “76000-92000”,
		“approvers”: [
		7
		],
		“owners”: [
		7
		]
		}
		}
		
		OR for XML
		
		<root>
		<requisition>
		<numOpen>2</numOpen>
		<workflowDefId>13</workflowDefId
		<department>3</department>
		<description>Head Nurse</description>
		<jobCat>(1.2) First/Mid Level Officials & Managers</jobCat>
		<duration>FULLTIME</duration>
		<isTemplate>true</isTemplate>
		<jobCode>NURSE-1293</jobCode>
		<location>1</location>
		<openedDate>2008-12-02</openedDate>
		<payRange>76000-92000</payRange>
		<approvers>
		<item>7</item>
		</approvers>
		<owners>
		<item>7</item>
		</owners>
		<status>2</status>

		<title>Head Nurse</title> </requisition> </root>

POST	Upsert Requisition	…/object/requisition?upsert=true
		
		{
		“requisition”: {
		“reqId”: 297,
		“title”: “Head Nurse”,
		“status”: “2”,
		“numOpen”: “2”,
		“candReqApplication”: “”,
		“workflowDefId”: 13,
		“department”: 3,
		“description”: “Head Nurse”,
		“jobCat”: “(1.2) First/Mid Level Officials & Managers”,
		“duration”: “FULLTIME”,
		“isTemplate”: true,
		“jobCode”: “NURSE-1293”,
		“location”: 1,
		“offerApprs”: [],
		“openedDate”: “2008-12-02”,
		“payRange”: “76000-92000”,
		“approvers”: [
		7
		],
		“owners”: [
		7
		]
		}
		}
		
		OR for XML
		
		<root>
		<requisition>
		<reqId>297</reqId>
		<numOpen>2</numOpen>
		<workflowDefId>13</workflowDefId
		<department>3</department>
		<description>Head Nurse</description>
		<jobCat>(1.2) First/Mid Level Officials & Managers</jobCat>
		<duration>FULLTIME</duration>
		<isTemplate>true</isTemplate>
		<jobCode>NURSE-1293</jobCode>
		<location>1</location>
		<openedDate>2008-12-02</openedDate>
		<payRange>76000-92000</payRange>
		<approvers>
		<item>7</item>

		</approvers> <owners> <item>7</item> </owners> <status>2</status> <title>Head Nurse</title> </requisition> </root>

PUT	Update Requisition	…/object/requisition/297
		
		{
		“requisition”: {
		“reqId”: 297,
		“filledDate”: “2008-12-02”,
		“status”: “4”,
		“numOpen”: “0”
		}
		}
		
		OR for XML
		
		<root>
		<requisition>
		<reqId>5</reqId>
		<status>5</status>
		</ requisition>
		</root>

WEB SERVICE REFERENCE: REQUISITION TEMPLATE ADMINISTRATION

The Taleo Business Edition requisition templates are used for storing Templates in the Job Library for creating Requisitions for recruiting. Requisition Templates can be managed through the Requisitions API by specifying the Boolean / checkbox field of isTemplate=true.

However, because Requisitions and Requisition Templates have differences in their required fields, the Requisition Template API can be used to manage the Requisition Templates separately.

With the Requisition Template API, the isTemplate=true is implied and need not be specified. In addition, there is no need to specify Owners nor Locations for the Requisition Templates.

To search for a requisition template, the URL format would be:

<<HOST_URL>>/object/requisitiontemplate/search?

When searching through requisition templates, all Integer, Date and Currency fields should include Parameters <fieldname>_from and/or <fieldname>_to. Please refer to the URL Structure section of this document for full list of parameters and other notes on searching.

The field definitions for requisition templates are available through the following discovery calls (standard fields only through the first URL vs including custom fields through the second provided URL):

<<HOST_URL>>/object/requisitiontemplate/description/**standard**

<<HOST_URL>>/ object/ requisitiontemplate/description/**custom**

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Requisition Templates		?	×	?	?	?	?
reqId	Integer	Y		Y	N	N	Y (Delete record by ID)
actions	Table Column	N		N	N	N	
addedWithin	Picklist	N		Y	N	N	
approvers	Custom	Y		Y	Y	Y	
Budgeted	Picklist	Y		Y	Y	Y	
city	Text	Y		Y	Y	Y	
creationDate	Date	N		Y	N	N	
currUserApprStatus	Check Box	N		N	N	N	
currUserDept	Table Column	N		N	N	N	
currUserDivision	Table Column	N		N	N	N	
currUserLocation	Table Column	N		N	N	N	
currUserOnlyOwner	Table Column	N		N	N	N	
currUserOwner	Table Column	N		N	N	N	
currUserRegion	Table Column	N		N	N	N	
currUserRegionLoc	Table Column	N		N	N	N	
cws	Picklist	N		Y	N	N	
department	Picklist	Y		Y	Y	Y	
description	Text Area	Y		Y	Y	Y	
division	Picklist (multiple)	Y		Y	Y	Y	
duration	Picklist	Y		Y	Y	Y	
eQuest Posting Site	Picklist	Y		Y	Y	Y	
filledDate	Date	Y		Y	Y	Y	
isHotJob	Check Box	Y		Y	Y	Y	
JobReason	Picklist	Y		Y	Y	Y	
JobBrief	Text(255)	Y		N	Y	Y	
jobCat	Picklist	Y		Y	Y	Y	
jobCode	Text(20)	Y		Y	Y	Y	
jobFunction	Picklist	Y		Y	Y	Y	
jobIndustry	Picklist	Y		Y	Y	Y	
jobType	Picklist	Y		Y	Y	Y	
keywords	Text	N		N	N	N	
lastUpdated	Date	Y		Y	N	N	
location	Picklist	Y		Y	Y	Y	
numCands	Integer	N		Y	N	N	
numOpen	Integer	Y		Y	Y	Y	
offerApprs	Custom	Y		Y	Y	Y	
openedDate	Date	Y		Y	Y	Y	
owners	Custom	Y		Y	Y	Y	
payRange	Text(100)	Y		Y	Y	Y	
region	Picklist (multiple)	Y		Y	Y	Y	
relevance	Table Column	N		N	N	N	

Replacement for	Text	Y		Y	Y	Y	
reqCandAce	Check Box	N		Y	N	N	
reqCandDateApplied	Date	N		Y	N	N	
reqCandidatesIcon	Requisition Candidates Icon	N		N	N	N	
reqCandNextSteps	Table Column	N		N	N	N	
reqCandRank	Integer	N		Y	N	N	
reqCandResRej	Picklist (multiple)	N		Y	N	N	
reqCandStatus	Picklist	N		Y	N	N	
reqCandStatusAction	Table Column	N		N	N	N	
reqHiringManager	Custom	Y		Y	Y	Y	
reqRecruiter	Custom	Y		Y	Y	Y	
state	Picklist	Y		Y	Y	Y	
status	Picklist	Y		Y	**Y***	Y	
title	Text(150)	Y		Y	**Y***	Y	
txReqStatus	Partner field	N		N	N	N	
updatedWithin	Picklist	N		Y	N	N	
viewers	Custom	N		N	N	N	
workflowDefId	Picklist	Y		Y	Y	Y	
zipCode	Text	Y		Y	Y	Y	
ebsRequisitionNumber	**Text(100)** *** If enabled in TBE**	**Y**		**Y**	**Y**	**Y**	
ebsIntegrationStatus	Picklist * If enabled in TBE	Y		Y	Y	Y	
fusionRequisitionNumber	Text(100) * If enabled in TBE	Y		Y	Y	Y	
fusionIntegrationStatus	Picklist * If enabled in TBE	Y		Y	Y	Y	
fusionBusinessUniteId	Picklist * If enabled in TBE	Y		Y	Y	Y	
fusionLegalEntityId	Picklist * If enabled in TBE	Y		Y	Y	Y	
jdeRequisitionNumber	Text(100) * If enabled in TBE	Y		Y	Y	Y	
jdeIntegrationStatus	Picklist * If enabled in TBE	Y		Y	Y	Y	
psoftRequisitionNumber	Text(100) * If enabled in TBE	Y		Y	Y	Y	
psoftIntegrationStatus	Picklist * If enabled in TBE	Y		Y	Y	Y	
reqSetHiredDate	Check Box	Y		N	Y	Y	
reqDecrementOpenings	Check Box	Y		N	Y	Y	
reqEmailNonHired	Check Box	Y		N	Y	Y	
reqChangeStatusNonHired	Check Box	Y		N	Y	Y	
reqChangeStatusPosting	Check Box	Y		N	Y	Y	
*** CUSTOM FIELDS		Y		Y	Y	Y	

Y* = required for creation.

Relationship URLs:

Relationship	URL
Department	…/object/requisitiontemplate/<reqId>/department
Location	…/object/requisitiontemplate /<reqId>/location
Offer Approvers	…/object/requisitiontemplate/reqId>/offerApprs
Regions	…/object/requisitiontemplate/<reqId>/region
Req Approvers	…/object/requisitiontemplate/<reqId>/approvers

Owners	…/object/requisitiontemplate/<reqId>/owners
Status	…/object/requisitiontemplate/<reqId>/status
Attachments	…/object/requisitiontemplate/<reqId>/attachment
Candidates	…/object/requisitiontemplate/<reqId>/candidate
Expense	…/object/requisitiontemplate/<reqId>/expense
Comment	…/object/requisitiontemplate/<reqId>/comment
Question	…/object/requisitiontemplate/<reqId>/question
History Log	…/object/requisitiontemplate/<reqId>/historylog

Sample Code Requisition Templates

Method	Action	Sample Request
GET	Get Requisition Template by ID	…/object/requisitiontemplate/69
GET	Search for Requisition Template	…/object/requisitiontemplate/search?status=2&duration=FULLTIME
DELETE	Delete Requisition Template by ID	…/object/requisitiontemplate/69

POST	Create Requisition Template	…/object/requisitiontemplate
		
		{
		“requisitiontemplate”: {
		“title”: “Head Nurse”,
		“status”: “2”,
		“numOpen”: “2”,
		“candReqApplication”: “”,
		“workflowDefId”: 13,
		“department”: 3,
		“description”: “Head Nurse”,
		“jobCat”: “(1.2) First/Mid Level Officials & Managers”,
		“duration”: “FULLTIME”,
		“jobCode”: “NURSE-1293”,
		“offerApprs”: [],
		“openedDate”: “2008-12-02”,
		“payRange”: “76000-92000”,
		“approvers”: [
		7
		],
		}
		}
		
		OR for XML
		
		<root>
		<requisitiontemplate>
		<numOpen>2</numOpen>
		<workflowDefId>13</workflowDefId
		<department>3</department>
		<description>Head Nurse</description>
		<jobCat>(1.2) First/Mid Level Officials & Managers</jobCat>
		<duration>FULLTIME</duration>
		<jobCode>NURSE-1293</jobCode>

		<openedDate>2008-12-02</openedDate> <payRange>76000-92000</payRange> <approvers> <item>7</item> </approvers> <status>2</status> <title>Head Nurse</title> </requisitiontemplate> </root>

POST	Upsert Requisition Template	…/object/requisitiontemplate?upsert=true
		
		{
		“requisitiontemplate”: {
		“reqId”: 297,
		“title”: “Head Nurse”,
		“status”: “2”,
		“numOpen”: “2”,
		“candReqApplication”: “”,
		“workflowDefId”: 13,
		“department”: 3,
		“description”: “Head Nurse”,
		“jobCat”: “(1.2) First/Mid Level Officials & Managers”,
		“duration”: “FULLTIME”,
		“jobCode”: “NURSE-1293”,
		“offerApprs”: [],
		“openedDate”: “2008-12-02”,
		“payRange”: “76000-92000”,
		“approvers”: [
		7
		]
		}
		}
		
		OR for XML
		
		<root>
		<requisitiontemplate>
		<reqId>297</reqId>
		<numOpen>2</numOpen>
		<workflowDefId>13</workflowDefId
		<department>3</department>
		<description>Head Nurse</description>
		<jobCat>(1.2) First/Mid Level Officials & Managers</jobCat>
		<duration>FULLTIME</duration>
		<jobCode>NURSE-1293</jobCode>
		<openedDate>2008-12-02</openedDate>
		<payRange>76000-92000</payRange>
		<approvers>
		<item>7</item>
		</approvers>
		<status>2</status>
		<title>Head Nurse</title>
		</requisitiontemplate>

		</root>

PUT	Update Requisition Template	…/object/requisitiontemplate/297
		
		{
		“requisitiontemplate”: {
		“reqId”: 297,
		“filledDate”: “2008-12-02”,
		“status”: “4”,
		“numOpen”: “0”
		}
		}
		
		OR for XML
		
		<root>
		<requisitiontemplate>
		<reqId>5</reqId>
		<status>5</status>
		</ requisitiontemplate>
		</root>

WEB SERVICE REFERENCE: REQUISITION APPROVALS MANAGEMENT

See “OFFER APPROVALS MANAGEMENT” Section.

WEB SERVICE REFERENCE: CAREER WEBSITES

Customers will be able to discover, using the REST APIs, the current set up of their Career Website(s) in Oracle Taleo Business Edition. For example, customers could retrieve the RSS URL for a Career Website or whether RSS Feed is enabled.

NOTE: This resource supports only search/retrieval. Career Websites cannot be created nor updated via the REST APIs.

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Career Website		✔	✔	✔	×	×	×
id	Integer	Y		N			
enableSocialSourcing	Check Box	Y		Y			
showRss	Check Box	Y		Y			
showFacebook	Check Box	Y		Y			
name	Text	Y		Y			
source	Text	Y		N			
url	Text	Y		N			
rssUrl	Text	Y		N			
Language	Text	Y		N			

Sample Code Careers Website

Method	Action	Sample Request

GET	Get CWS by ID	…/object/careerswebsite/<id>
GET	Get all CWSs	…/object/careerswebsite/search
GET	Search CWSs	…/object/careerswebsite/search?enableSocialSourcing=false

WEB SERVICE REFERENCE: REQUISITION POSTING ADMINISTRATION

Please Note: There is no JSON hyperschema support for Requisition Posting Administration.

The Taleo Business Edition Requisition Poster resource can be used to search and manage the posting status of a requisition on the Career Websites.

The Poster REST API Resource allows the customers to discover the current status of the Postings for a Requisition. For example, on which Career Websites a Requisition is currently posted, if a requisition is currently posted to a particular CWS or when it is scheduled to be delisted.

The Poster resource also supports the POST method to post Requisitions to particular CWSs of update their posting dates/forms etc.

A Requisition posting can be removed from a Career Website all together, the URL for the posting can be used with the DELETE method.

Note: The requisition ID of the requisition the posting(s) are for must be known and included in the URL.

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Poster		✔	✔	✔	✔	✔	✔
reqId	Integer	Y		N	Y	N	Y (in URL)
careerWebSiteId	Integer	Y		N	**Y***	N	Y
formVersion	Integer	Y		N	**Y***	Y	
postDate	Date	Y		N	**Y***	Y	
delistDate	Date	Y		N	**Y***	Y	
url	Text	Y		N	N	N	

Y* = required for creation.

Sample Code Requisition Poster

Method	Action	Sample Request
GET	Get all postings for a requisition.	…/object/requisition/<reqId>/poster Or …/object/requisition/<reqId>/poster.xml
GET	Get a posting for a requisition.	…/object/requisition/<reqId>/poster/<careerWebSiteId> Or …/object/requisition/<reqId>/poster/<careerWebSiteId>.xml

DELETE	Remove a Requisition Posting from a CWS Will remove posting for requisition (ReqID) from Career Webstie (CareerWebSiteId).	…/object/requisition/<reqId>/poster/<careerWebSiteId> Or …/object/requisition/<reqId>/poster/<careerWebSiteId>.xml

POST	Create Requisition Template
	
	Post Requisition (ReqId in URL) to the CWS (careerWebSiteId).
	
	NOTE: All fields are currently required for create.
		…/object/requisitiontemplate
		
		{
		“poster “: {
		“ careerWebSiteId “: “37”,
		“ formVersion “: “0”,
		“ 111vailabl”: “2013-09-24”,
		“ delistDate”: “2019-09-24”,
		“workflowDefId”: 13
		}
		}
		
		OR for XML
		
		…/object/requisition/<reqId>/poster.xml
		
		<root>
		<poster>
		<careerWebSiteId>37</careerWebSiteId>
		<formVersion>0</formVersion>
		<111vailabl>2013-09-24</111vailabl>
		<delistDate>2019-09-24</delistDate>
		</poster>
		</root>

POST	Upsert Requisition Template
	
	Post Requisition (ReqId in URL) to the CWS (careerWebSiteId).
	
	NOTE: All fields are currently required for create.
		…/object/requisitiontemplate?upsert=true
		
		{
		“poster “: {
		“ careerWebSiteId “: “37”,
		“ formVersion “: “0”,
		“ 111vailabl”: “2013-09-24”,
		“ delistDate”: “2019-09-24”,
		“workflowDefId”: 13
		}
		}
		
		OR for XML
		
		…/object/requisition/<reqId>/poster.xml?upsert=true
		
		<root>
		<poster>
		<careerWebSiteId>37</careerWebSiteId>
		<formVersion>0</formVersion>
		<111vailabl>2013-09-24</111vailabl>
		<delistDate>2019-09-24</delistDate>
		</poster>

		</root>

WEB SERVICE REFERENCE: QUESTIONS LIBRARY

The Taleo Business Edition Questions Library is used for storing screening questions which can be attached to each requisition and presented to candidates when they apply online. Although we do not support the association of questions to requisitions via the API, the maintenance of the library is possible through the API.

We also support the management of answers to these questions.

Relationship URLs:

Relationship	URL
Answers	…/object/question/<questId>/answer

The following code provides information for both questions and answers

Method	Action	Sample Request
GET	Retrieves metadata	…/object/info/question
GET	Retrieves standard fields – hardcoded	…/object/question/description/standard
GET	Retrieves display field info – hardcoded	…/object/displayfield/QUEST/{fieldname}
GET	Retrieves questions by category; returns all questions if no category is passed	…/object/question(?category={category})
GET	Retrieves question by id	…/object/question/{id}
GET	Retrieves all questions related to requisition	…/object/requisition/{id}/question
DELETE	Deletes question	…/object/question/{id}
POST	Creates question	…/object/question
POST	Upserts question by id	…/object/question/{id}

	Sample Question JSON	{
		“question”: {
		“category”: “General”,
		“type”: “T”,
		“question”: “Second REST question”
		}
		}

GET	Retrieves metadata	…/object/info/question/answer
GET	Retrieves standard fields – hardcoded	…/object/question/answer/description/standard
GET	Retrieves custom fields – same as standard	…/object/question/answer/description/custom
GET	Retrieves display field info – hardcoded	…/object/displayfield/ANSWE/{112vailable}
GET	Retrieves all answers for question	…/object/question/{id}/answer
GET	Retrieves answer for question by seqNo	…/object/question/{id}/answer/{seqNo}

DELETE	Deletes answer	…/object/question/{id}/answer/{seqNo}
POST	Creates answer for question	…/object/question/{id}/answer
PUT	Upserts answer by seqNo	…/object/question/{id}/answer/{seqNo}

	Sample Answer JSON	{
		“answer”: {
		“seqNo”: 0,
		“answer”: “REST question first answer”,
		“weight”: 10,
		“answerCriteria”: “ACE”
		}
		}

WEB SERVICE REFERENCE: CANDIDATE INTERVIEWS ADMINISTRATION

The Taleo Business Edition interview table is a relationship table to candidates for collection of interview scheduling details. This includes the candidate ID, core interview information and an array of participants that are tied to the interview.

To receive a single interview, use the following URL:

<<HOST_URL>>/object/interview/ **<<Interview_ID>>**

For all interviews associated with a single candidate, use the following URL:

<<HOST_URL>>/object/candidate/**<<Candidate_ID>>**/interview/

The field definitions for interviews are available through the following discovery calls (standard fields only through the first URL vs including custom fields):

<<HOST_URL>>/object/interview/description/**standard**

<<HOST_URL>>/ object/interview/description/**custom**

The JSON hyperschema field definitions for interviews are available through the following discovery call):

<<HOST_URL>>/object/**interview**/hyperschema.json

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Interviews		✔	×	×	✔	✔	✔
intvId	Integer	Y			N	N	Y (record by ID)
actions	Table Column	N			N	N	
candidate	Custom	Y			Y	Y	
comment	Text Area	Y			Y	Y	
creationDate	Date	Y			N	N	
date	Date	N			N	N	
dateTime	Custom	Y			Y	Y	
feedbackTempl	Picklist	Y			Y	Y	
intvRoom	Picklist	Y			Y	Y	
lastUpdated	Date	Y			N	N	
location	Text	Y			Y	Y	
myStatus	Custom	N			N	N	
requisition	Picklist	Y			Y	Y	
scheduler	Text	Y			Y	Y	

status	Picklist	Y	Y	Y
time	Time	N	N	N
Interview	Picklist	Y	Y	Y
participants	Custom (Multiple)	Y	N	N
participant.personId	User	Y		
participant.personType	Entity	Y		
participant.eventId	Custom	Y		
participant.eventType	Custom	Y		
participant.startDate	DateTime	Y		
participant.endDate	DateTime	Y		
***** CUSTOM FIELDS**		Y	Y	Y

Relationship URLs:

Relationship	URL
Candidate	…/object/interview/<intvId>/candidate
Location	…/object/interview/<intvId>/location
Requisition	…/object/interview/<intvId>/requisition
Scheduler	…/object/interview/<intvId>/scheduler
Status	…/object/interview/<intvId>/status
Interview Feedbacks	…/object/interview/<intvId>/interviewfeedback
History Log	…/object/interview/<intvId>/historylog
Participant Relationship URLs	
Person	…/object/user/<personId>
Paticipant Feedback	…/object/interview/<intvId>/participant/<personId>/interviewfeedback

Sample Code Interviews

Please Note: When creating an interview via the API, an interview feedback session will be created in cognizant. Get the interview via response URL to find the correlated interview feedback record.

When a timezone is not provided on the dateTime field for an interview session, it will default the locale of the API user who is making the call.

Method	Action	Sample Request
GET	Get Single Interview by ID	…/object/interview/1
GET	Get all interviews tied to a candidate ID	…/object/candidate/57/interview
DELETE	Delete Interview by ID	…/object/interview/60

POST	Create Account	…/object/interview
		
		{
		“interview”: {
		“candidate”: 40,
		“comment”: “TEST Interview”,
		“dateTime”: “2010-09-14T08:00PDT”,
		“feedbackTempl”: 1,
		“intvRoom”: “Offices”,
		“location”: 1,
		“requisition”: 65,
		“scheduler”: 15,

		“status”: 2,
		“Interview”: “In-person”,
		“participants”: [
		{
		“participant”: {
		“personId”: 42,
		“personType”: “WORK”,
		“eventType”: “INTV”,
		“startDate”: “2010-09-14T08:00PDT”,
		“endDate”: “2010-09-14T08:30PDT”
		}
		}
		]
		}
		}
		
		OR for XML
		
		<root>
		<interview>
		<candidate>57</candidate>
		<comment>TEST</comment>
		<dateTime>2010-09-14T08:00PDT</dateTime>
		<feedbackTempl>1</feedbackTempl>
		<intvRoom>Offices</intvRoom>
		<location>1</location>
		<requisition>65</requisition>
		<scheduler>15</scheduler>
		<status>2</status>
		<Interview>In-person</Interview>
		<participants>
		<item>
		<participant>
		<personId>42</personId>
		<personType>WORK</personType>
		<eventType>INTV</eventType>
		<startDate>2010-09-14T08:00PDT</startDate>
		<endDate>2010-09-14T08:30PDT</endDate>
		</participant>
		</item>
		</participants>
		</interview>
		</root>

POST	Upsert Interview	…/object/interview?upsert=true
		
		{
		“interview”: {
		“intvId”: 60,
		“candidate”: 40,
		“comment”: “TEST Interview”,

		“dateTime”: “2010-09-14T08:00PDT”,
		“feedbackTempl”: 1,
		“intvRoom”: “Offices”,
		“location”: 1,
		“requisition”: 65,
		“scheduler”: 15,
		“status”: 2,
		“Interview”: “In-person”,
		“participants”: [
		{
		“participant”: {
		“personId”: 42,
		“personType”: “WORK”,
		“eventType”: “INTV”,
		“startDate”: “2010-09-14T08:00PDT”,
		“endDate”: “2010-09-14T08:30PDT”
		}
		}
		]
		}
		}
		
		OR for XML
		
		<root>
		<interview>
		<intvId>60</intvId >
		<candidate>57</candidate>
		<comment>TEST</comment>
		<dateTime>2010-09-14T08:00PDT</dateTime>
		<feedbackTempl>1</feedbackTempl>
		<intvRoom>Offices</intvRoom>
		<location>1</location>
		<requisition>65</requisition>
		<scheduler>15</scheduler>
		<status>2</status>
		<Interview>In-person</Interview>
		<participants>
		<item>
		<participant>
		<personId>42</personId>
		<personType>WORK</personType>
		<eventType>INTV</eventType>
		<startDate>2010-09-14T08:00PDT</startDate>
		<endDate>2010-09-14T08:30PDT</endDate>
		</participant>
		</item>
		</participants>
		</interview>
		</root>

PUT	Update Interview	…/object/interview/60
		
		{
		“interview”: {
		“comment”: “Great Interview”
		}
		}
		
		OR for XML
		
		<root>
		<interview>
		<intvId>60</intvId >
		<participants>
		<item>
		<participant>
		<personId>42</personId>
		<personType>WORK</personType>
		<eventType>INTV</eventType>
		<startDate>2010-09-14T08:00PDT</startDate>
		<endDate>2010-09-14T08:30PDT</endDate>
		</participant>
		</item>
		</participants>
		</interview>
		</root>

WEB SERVICE REFERENCE: CANDIDATE INTERVIEW FEEDBACK

The Taleo Business Edition interview feedback table is connected to an interview instance, specifically to an interview participant. It allows for a participant to file feedback for an interview and have that connected to a candidate for assessment evaluation.

To receive a single interview feedback item, use the following URL:

<<HOST_URL>>/object/interviewfeedback/ **<<Feed_ID>>**

For all interview feedbacks associated with a single interview, use the following URL:

<<HOST_URL>>/object/interview/**<<Interview_ID>>**/interviewfeedback/

The field definitions for interview feedback is available through the following discovery calls (standard fields only through the first URL vs including custom fields):

<<HOST_URL>>/object/interviewfeedback/description/**standard**

<<HOST_URL>>/ object/interviewfeedback/description/**custom**

The JSON hyperschema field definitions for interview feedback are available through the following discovery call):

<<HOST_URL>>/object/**interviewfeedback**/hyperschema.json

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Interviews		✔	✔	×	✔	✔	✔
feedId	Integer	Y	Y		Y	Y	Y (record by ID)
actions	Table Column	N	N		N	N	
candidate	Custom	Y	Y		Y	Y	
comment	Text Area	Y	Y		Y	Y	
Interview	Custom	Y	Y		Y	Y	
lastUpdated	Date	Y	Y		N	N	
Summary	Picklist	Y	Y		Y	Y	
requisition	Custom	Y	Y		Y	Y	
date	Date	Y	Y		Y	Y	
status	Picklist	Y	Y		Y	Y	
creationDate	Date	Y	Y		N	N	
submitter	User	Y	Y		Y	Y	
***** CUSTOM FIELDS**		Y	Y		Y	Y	

Relationship URLs:

Relationship	URL
Candidate	…/object/interviewfeedback/<feedId>/candidate
Interview	…/object/interviewfeedback/<feedId>/Interview
Requisition	…/object/interviewfeedback/<feedId>/requisition
Status	…/object/interviewfeedback/<feedId>/status
Submitter	…/object/interviewfeedback/<feedId>/submitter

Sample Code Interview Feedbacks

Method	Action	Sample Request
GET	Get Single Interview feedback by ID	…/object/interviewfeedback/1
GET	Get all interview feedback tied to a single interview	…/object/interview/57/interviewfeedback
DELETE	Delete Interview Feedback by ID	…/object/interviewfeedback/60

POST	Create An Interview Feedback	…/object/interviewfeedback
		
		{
		“interviewfeedback”: {
		“candidate”: 57,
		“comment”: “Great Fit”,
		“Interview”: 60,
		“Summary”: “Yes”,
		“requisition”: 65,
		“date”: “2010-09-14T08:00PDT”,
		“status”: 5,
		“submitter”: 11
		}
		}
		
		OR for XML

		<root> <interviewfeedback> <candidate>57</candidate> <comment>Great Fit</comment> <Interview>60</Interview> <Summary>Yes</Summary> <requisition>65</requisition> <date>2010-09-14T08:00PDT</date> <status>5</status> <submitter>11</submitter> </interviewfeedback> </root>

POST	Upsert Interview	…/object/interviewfeedback?upsert=true
		
		{
		“interviewfeedback”: {
		“candidate”: 57,
		“comment”: “Great Fit”,
		“Interview”: 60,
		“Summary”: “Yes”,
		“requisition”: 65,
		“date”: “2010-09-14T08:00PDT”,
		“status”: 5,
		“feedId”:63,
		“submitter”: 11
		}
		}
		
		OR for XML
		
		<root>
		<interviewfeedback>
		<feedId>63</feedId>
		<candidate>57</candidate>
		<comment>Great Fit</comment>
		<Interview>60</Interview>
		<Summary>Yes</Summary>
		<requisition>65</requisition>
		<date>2010-09-14T08:00PDT</date>
		<status>5</status>
		<submitter>11</submitter>
		</interviewfeedback>
		</root>

PUT	Update Interview Feedback	…/object/interviewfeedback/63
		
		{
		“interviewfeedback”: {
		“candidate”: 57,
		“comment”: “Great Fit”,

		“Interview”: 60, “Summary”: “Yes”, “requisition”: 65, “date”: “2010-09-14T08:00PDT”, “status”: 5, “feedId”:63, “submitter”: 11 } } **OR for XML** <root> <interviewfeedback> <feedId>63</feedId> <candidate>57</candidate> <comment>Great Fit</comment> <Interview>60</Interview> <Summary>Yes</Summary> <requisition>65</requisition> <date>2010-09-14T08:00PDT</date> <status>5</status> <submitter>11</submitter> </interviewfeedback> </root>

WEB SERVICE REFERENCE: CANDIDATE BACKGROUND CHECKS

The Taleo Business Edition background check table is connected to a candidate record for historic logging of assessments and screens.

The field definitions for candidate background checks are available through the following discovery calls (standard fields only through the first URL vs including custom fields):

<<HOST_URL>>/object/backgroundcheck/description/**standard**

<<HOST_URL>>/object/backgroundcheck/description/**custom**

The JSON hyperschema field definitions for candidate background checks are available through the following discovery call):

<<HOST_URL>>/object/**backgroundcheck**/hyperschema.json

To receive a single background check item, use the following URL:

<<HOST_URL>>/object/backgroundcheck/**<<backgroundcheck_ID>>**

For all background checks associated with a single candidate, use the following URL:

<<HOST_URL>>/object/candidate/**<<Candidate_ID>>**/backgroundcheck/

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
backgroundcheck		✔	✔	×	✔	✔	✔

bkgrId	Integer	Y	Y	Y	Y	Y (record by ID)
candidate	Custom	N	N	N	N	
comment	Text Area	Y	Y	Y	Y	
creationDate	Date	Y	Y	N	N	
email	Email	Y	Y	Y	Y	
lastUpdated	Date	Y	Y	N	N	
checkName	Text(80)	Y	Y	Y	Y	
phone	Phone	Y	Y	Y	Y	
userId	User	Y	Y	Y	Y	
*** **CUSTOM FIELDS**		Y	Y	Y	Y	

Relationship URLs:

Relationship	URL
Candidate	…/object/backgroundcheck/<bkgrId>/candidate
User	…/object/backgroundcheck/<bkgrId>/user

Sample Code Background Checks

Method	Action	Sample Request
GET	Get Single Background Check by ID	…/object/backgroundcheck/1
GET	Get all Background Checks tied to a single candidate	…/object/candidate/57/backgroundcheck
DELETE	Delete Background Check by ID	…/object/backgroundcheck/60

POST	Create a Background Check	…/object/backgroundcheck
		
		{
		“backgroundcheck”: {
		“candidate”: 57,
		“comment”: “Passed Test”,
		“email”: “skarim@taleo.com”,
		“checkName”: “Sample”,
		“phone”: “293-123-13934”,
		“userId”: 11
		}
		}
		
		OR for XML
		
		<root>
		<backgroundcheck>
		<candidate>57</candidate>
		<comment>Passed Test</comment>
		<email>skarim@taleo.com</email>
		<checkName>Sample</checkName>
		<phone>293-123-13934</phone>
		<userId>11</userId>
		</backgroundcheck>
		</root>

POST	Upsert Background Check	…/object/ backgroundcheck?upsert=true
		
		{
		“backgroundcheck”: {
		“candidate”: 57,
		“ bkgrId “: 57,
		“comment”: “Passed Test”,
		“email”: “skarim@taleo.com”,
		“checkName”: “Sample”,
		“phone”: “293-123-13934”,
		“userId”: 11
		}
		}
		
		OR for XML
		
		<root>
		<backgroundcheck>
		<bkgrId>34</bkgrId>
		<candidate>57</candidate>
		<comment>Passed Test</comment>
		<email>skarim@taleo.com</email>
		<checkName>Sample</checkName>
		<phone>293-123-13934</phone>
		<userId>11</userId>
		</backgroundcheck>
		</root>

PUT	Update Background Check	…/object/backgroundcheck/63
		
		{
		“backgroundcheck”: {
		“comment”: “Passed Test”,
		}
		}
		
		OR for XML
		
		<root>
		<backgroundcheck>
		<bkgrId>34</bkgrId>
		<candidate>57</candidate>
		<comment>Passed Test</comment>
		<email>skarim@taleo.com</email>
		<checkName>Sample</checkName>

		<phone>293-123-13934</phone> <userId>11</userId> </backgroundcheck> </root>

WEB SERVICE REFERENCE: RECRUITING EXPENSES MANAGEMENT

Taleo Business Edition allows logging of expenses for requisitions or candidates.

When logging or retrieving an expense specific to a requisition solely, these are the fields you will be engaged with:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Expense		✔	✔	×	✔	✔	✔
expenseId	Integer	Y	Y		N	N	Y (record by ID)
requisition	Integer	Y	Y		Y	Y	
amount	Currency	Y	Y		Y	Y	
user	Integer	Y	Y		Y	Y	
description	Text(100)	Y	Y		Y	Y	
source	Picklist	Y	Y		Y	Y	

When logging or retrieving an expense that involves candidates, these are the fields you will be engaged with:

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Expense		✔	✔	×	✔	✔	✔
expenseId	Integer	Y	Y		N	N	Y (record by ID)
requisition	Integer	Y	Y		Y	Y	
candidate	Integer	Y	Y		Y	Y	
category	Picklist	Y	Y		Y	Y	
amount	Currency	Y	Y		Y	Y	
user	Integer	Y	Y		Y	Y	
description	Text(100)	Y	Y		Y	Y	

Relationship URLs:

Relationship	URL
Candidate	…/object/expense/<expenseId>/candidate
Requisition	…/object/expense/<expenseId>/requisition
User	…/object/expense/<expenseId>/user

Sample Code Expenses

Method	Action	Sample Request
GET	Get Single expense item	…/object/expense/1
GET	Get all expenses tied to a single candidate	…/object/candidate/57/expense
GET	Get all expenses tied to a single requisition	…/object/requisition/75/expense

DELETE	Delete Expense by ID	…/object/expense/60

POST	Create an Expense instance for Requisitions	…/object/expense
		
		{
		“expense”: {
		“amount”: “233.00”,
		“category”: “Job Board Posting”,
		“date”: “2012-04-22”,
		“description”: “Job Posting Fee”,
		“requisition”: 65,
		“source”: “HotJobs”
		}
		}
		
		OR for XML
		
		<root>
		<expense>
		<amount>233.00</amount>
		<category>Job Board Posting</category>
		<date>2012-04-22</date>
		<description>Job Posting Fee</description>
		<requisition>65</requisition>
		<source>HotJobs</source>
		</expense>
		</root>

POST	Upsert / Update Expense Instance for Candidate	…/object/expense?upsert=true
		
		{
		“expense”: {
		“amount”: “34.00”,
		“candidate”: 57,
		“category”: “Travel”,
		“date”: “2012-04-22”,
		“description”: “On Site interview”,
		“expenseId”: “282”,
		“requisition”: 65
		}
		}
		
		OR for XML
		
		<root>
		<expense>
		<amount>34.00</amount>
		<candidate>57</candidate>
		<category>Travel</category>
		<date>2012-04-22</date>
		<description>On Site interview</description>
		<expenseId>282</expenseId>
		<requisition>65</requisition>

		</expense> </root>

WEB SERVICE REFERENCE: RECRUIT ACCOUNT ADMINISTRATION

The Taleo Business Edition account table is used to store 3rd party organizations that support your recruiting efforts, for example staffing agencies and sourcing partners.

To search for an existing account, the URL format would be:

<<HOST_URL>>/object/account/search?

The field definitions for accounts are available through the following discovery calls (standard fields only through the first URL vs including custom fields):

<<HOST_URL>>/object/account/description/**standard**

<<HOST_URL>>/ object/account/description/**custom**

The JSON hyperschema field definitions for accounts are available through the following discovery call):

<<HOST_URL>>/object/**account**/hyperschema.json

Please Note: For Searching of Accounts, all Integer, Date and Currency fields should include Parameters <fieldName>_from and/or <fieldName>_to. Please refer to the URL Structure section of this document for full list of parameters and other notes on searching.

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Accounts		✔	×	✔	✔	✔	✔
accountId	Integer	Y		Y	N	N	Y
name	Text(100)	Y		Y	**Y***	Y	
owners	Account	Y		Y	Y	Y	
creationDate	Date	Y		Y	N	N	
addedWithin	Picklist	N		Y	N	N	
phoneAll	Phone	N		N	N	N	
city	Text(30)	N		N	N	N	
country	Picklist	Y		Y	Y	Y	
description	Text Area	N		N	Y	Y	
keywords	Text	Y		Y	Y	Y	
fax	Phone	Y		Y	Y	Y	
industry	Picklist	Y		Y	N	N	
lastUpdated	Date	Y		Y	N	N	
linkedInSearch	Web Link	Y		N	Y	Y	
mapQuest	Web Link	Y		N	Y	Y	
parentId	Account	Y		Y	Y	Y	
phone	Phone	Y	`	Y	Y	Y	
relevance	Table Column	N		N	N	N	
state	Picklist	Y		Y	Y	Y	
address	Text(100)	Y		Y	Y	Y	
AcctType	Picklist	Y		Y	Y	Y	
updatedWithin	Picklist	N		Y	N	N	
websiteURL	URL	Y		Y	Y	Y	
zipCode	Text(100)	Y		Y	Y	Y	

***** CUSTOM FIELDS**	Y	Y	Y	Y

Y* = required for creation.

Relationship URLs:

Relationship	URL
Owners	…/object/account/<accountId>/owners
Attachments	…/object/account/<accountId>/attachment
Comment	…/object/account/<accountId>/comment
Contact log	…/object/account/<accountId>/contactlog
History log	…/object/account/<accountId>/historylog

Sample Code Accounts

Please Note: By default the API outputs and assumes JSON. If XML is requested, please ensure adding the attribute of .xml to the end of the URL's in your request, including in the samples provided below.

For upsert parameter used to conduct a record update, even though you do not need to state the ID in the request URL, you will need to provide the accountId in the message body for update to match an existing record.

Method	Action	Sample Request
GET	Get Account by ID	…/object/account/1
GET	Search for Account	…/object/account/search?name=ITS%20Staffing
DELETE	Delete Account by ID	…/object/account/1

POST	Create Account	…/object/account
		
		{
		"account": {
		"name": "Test Account1",
		"owners": [
		15
		],
		"city": "Vatsonvile",
		"country": "US",
		"description": "Test Account1 description",
		"fax": "456-632-7412",
		"industry": "Aerospace",
		"parentId": 1,
		"phone": "415-124-5217",
		"state": "US-CA",
		"address": "475 New Line",
		"AcctType": "Search Firm",
		"websiteURL": "www.hello.com",
		"zipCode": "94215"
		}
		}
		
		OR for XML
		
		<root>
		<account>
		<name>Test Account1</name>

		<owners> <item>15</item> </owners> <city>Vatsonvile</city> <country>US</country> <description>Test Account1 description</description> <fax>456-632-7412</fax> <industry>Aerospace</industry> <parentId>1</parentId> <phone>415-124-5217</phone> <state>US-CA</state> <address>475 New Line</address> <AcctType>Search Firm</AcctType> <websiteURL>www.hello.com</websiteURL> <zipCode>94215</zipCode> </account> </root>

POST	Upsert Account	…/object/account?upsert=true
		
		{
		“account”: {
		“name”: “Test Account1”,
		“owners”: [
		15
		],
		“city”: “Vatsonvile”,
		“country”: “US”,
		“description”: “Test Account1 description”,
		“fax”: “456-632-7412”,
		“industry”: “Aerospace”,
		“accountId”: 5,
		“parentId”: 1,
		“phone”: “415-124-5217”,
		“state”: “US-CA”,
		“address”: “475 New Line”,
		“AcctType”: “Search Firm”,
		“websiteURL”: “www.hello.com”,
		“zipCode”: “94215”
		}
		}
		
		OR for XML
		
		<root>
		<account>
		<name>Test Account1</name>
		<owners>
		<item>15</item>
		</owners>
		<city>Vatsonvile</city>
		<country>US</country>
		<description>Test Account1 description</description>
		<fax>456-632-7412</fax>

		<industry>Aerospace</industry> <accountId>5</accountId> <parentId>1</parentId> <phone>415-124-5217</phone> <state>US-CA</state> <address>475 New Line</address> <AcctType>Search Firm</AcctType> <websiteURL>www.hello.com</websiteURL> <zipCode>94215</zipCode> </account> </root>

PUT	Update Account	…/object/account/5
		
		{
		“account”: {
		“accountId”: 5,
		“parentId”: 1
		}
		}
		
		OR for XML
		
		<root>
		<account>
		<accountId>5</accountId>
		<parentId>1</parentId>
		</account>
		</root>

WEB SERVICE REFERENCE: CONTACT ADMINISTRATION

Please note the following supported RESTful actions for Contacts in Taleo Business Edition:

Please Note: For Searching of Contacts, all Integer, Date and Currency fields should include Parameters <fieldName>_from and/or <fieldName>_to. Please refer to the URL Structure section of this document for full list of parameters and other notes on searching.

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Contacts		✔	×	✔	✔	✔	✔
contactId	Integer	Y		Y	N	Y	Y
contTypeId	Picklist	Y		Y	Y	Y	
accountId	Account	Y		Y	Y	Y	
creationDate	Date	Y		Y	N	N	
addedWithin	Picklist	N		Y	N	N	
phoneAll	Phone	N		N	N	N	
asstName	Text(100)	Y		Y	Y	Y	
asstEmail	Email	Y		Y	Y	Y	
asstPhone	Phone	Y		Y	Y	Y	
city	Text(30)	Y		Y	Y	Y	

country	Picklist	Y	Y	Y	Y
userId	User	Y	N	N	N
department	Text(100)	Y	Y	Y	Y
description	Text Area	Y	Y	Y	Y
keywords	Text	N	N	N	N
email	Email	Y	Y	Y	Y
fax	Phone	Y	Y	Y	Y
firstName	Text(30)	Y	Y	Y	Y
googleSearch	Web Link	Y	N	N	N
lastName	Text(30)	Y	Y	**Y***	Y
lastUpdated	Date	Y	Y	N	N
linkedInSearch	Web Link	Y	N	N	N
mapQuest	Web Link	Y	N	N	N
129vailable	Phone	Y	Y	Y	Y
fullName	Text	N	N	N	N
contEmailRequired	Check Box	Y	Y	Y	Y
phone	Phone	Y	Y	Y	N
preferredLocale	Picklist	Y	N	Y	Y
relevance	Table Column	N	N	N	N
reptToId	Contact	Y	N	Y	Y
state	Picklist	Y	Y	Y	Y
address	Text(100)	Y	Y	Y	Y
title	Text(100)	Y	Y	Y	Y
updatedWithin	Picklist	Y	Y	N	N
zipCode	Text(10)	Y	Y	Y	Y
*** **CUSTOM FIELDS**		Y	Y	Y	Y

Y* = required for creation.

Relationship URLs:

Relationship	URL
User	…/object/contact/<contactId>/userId
Account	… /object/contact/<contactId>/account
Reports To	…/object/contact/<129vailable>/reportsTo
Comment	…/object/contact/<129vailable>/comment
Contact Log	…/object/contact/<129vailable>/contactlog

Sample Code Contacts

Please Note: By default the API outputs and assumes JSON. If XML is requested, please ensure adding the attribute of .xml to the end of the URL's in your request, including in the samples provided below.

For upsert parameter used to conduct a record update, even though you do not need to state the ID in the request URL, you will need to provide the contactID in the message body for update to match an existing record.

For update you can only send back the specific fields you are interested in updating.

Method	Action	Sample Request
GET	Get Contact by ID	…/object/contact/4 **OR for XML**

		…/object/contact/4.xml
GET	Search for contact	…/object/contact/search?start=1&limit=1&addedWithin=5000 **OR for XML** …/object/contact/search.xml?start=1&limit=1&addedWithin=5000
DELETE	Delete Contact by ID	…/object/contact/1 **OR for XML** …/object/contact/1.xml

POST	Create Contact	…/object/contact
		
		{
		“user”:{
		“loginName”:”time1323813641”,
		“lastName”:”webapikztfcnvgqzbdhnyfryjx”,
		“role”:”Administrator”,
		“status”:1,
		“email”:”time1323813641@tberegressionautomation.com”
		}
		}
		
		OR for XML
		
		<root>
		<user>
		<loginName>time1323813641</loginName>
		<lastName>webapibrrwafrtembkhfdfndng</lastName>
		<role>Administrator</role>
		<status type=”integer”>1</status>
		<email>time1323813641@tberegressionautomation.com</email>
		</user>
		</root>

POST	Upsert User	…/object/user?upsert=true
		
		{
		“user”:{
		“loginName”:”time1323975161”,
		“lastName”:”webapihjgazdpacskgkcesmchq”,
		“role”:”Administrator”,
		“userId”:568,
		“status”:1,
		“email”:”time1323975161@tbeautomation.com”
		}
		}
		
		OR for XML
		
		<root>
		<user>
		<userId>568</userId>

		<loginName>UPSERTtest</loginName> <lastName>upsertTEST</lastName> <role>Administrator</role> <status type=”integer”>2</status> <email>time1323813641@tasdfberegressionautomation.com</email> </user> </root>

PUT	Update User	…/object/user/568
		
		{
		“user”:{
		“AdminNewsletter”:true,
		“assignReviewAccess”:true,
		“submittedEmail”:true,
		“mainStatusContrAccss”:false,
		“compApprover”:true,
		“compContributor”:false,
		“compManager”:true,
		“confidentialAccess”:true,
		“newTmplAccess”:true,
		“newRevTempAccess”:false,
		“deleteRevTempAccess”:true,
		“editReviewAccess”:true,
		“editRevTempAccess”:true,
		“email”:”iguz123@taleo.com”,
		“firstName”:”igor1”,
		“GeneralNewsletter”:false,
		“jobLibAccess”:true,
		“lastName”:”guz”,
		“lastUpdated”:”9/29/11 10:25 AM”,
		“locale”:”enUS”,
		“location”:1,
		“manager”:11,
		“newReqAccess”:false,
		“offerAppr”:true,
		“reviewApprover”:false,
		“reviewManager”:true,
		“role”:”Administrator”,
		“status”:1,
		“timeZone”:”America/Los_Angeles”,
		“title”:”TEST”,
		“loginName”:”iguasdf1”,
		“viewReviewAccess”:true,
		}
		}
		
		OR for XML
		
		<root>
		<user>
		<AdminNewsletter>false</AdminNewsletter>
		<assignReviewAccess>true</assignReviewAccess>
		<mainStatusContrAccss>true</mainStatusContrAccss>
		<compApprover>true</compApprover>
		<compContributor>false</compContributor>
		<compManager>true</compManager>
		<confidentialAccess>true</confidentialAccess>
		<newTmplAccess>true</newTmplAccess>

		<newRevTempAccess>true</newRevTempAccess> <editPositionAccess>false</editPositionAccess> <editProfileAccess>true</editProfileAccess> <deleteRevTempAccess>true</deleteRevTempAccess> <editReviewAccess>true</editReviewAccess> <editRevTempAccess>true</editRevTempAccess> <newReqAccess>true</newReqAccess> <offerAppr>false</offerAppr> <reviewApprover>false</reviewApprover> <reviewManager>true</reviewManager> <approver>false</approver> <division> <item>49</item> </division> <department> <item>29</item> </department> <region> <item>1</item> </region> <email>iguz12asdfsad3@taleo.com</email> <employee>27</employee> <fax>23423542345</fax> <firstName>igor1</firstName> <lastName>guz</lastName> <locale>enUS</locale> <location>1</location> <manager>11</manager> <middleInitial></middleInitial> <132vailable></132vailable> <role>Administrator</role> <status>1</status> <timeZone>America/Los_Angeles</timeZone> <title></title> <loginName>iguasasdfasdf1</loginName> <viewReviewAccess>true</viewReviewAccess> <viewPositionAccess>false</viewPositionAccess> <viewProfileAccess>true</viewProfileAccess> </user> </root>

WEB SERVICE REFERENCE: CONTACT LOG ADMINISTRATION

The Taleo Business Edition Contact logs are a running trail of communication stored within a specific entity of Taleo Business Edition. There is a one-to-many relationship with contact logs to entities.

Entities that have associated Contact Logs are:

- Accounts (ACCT)
- Candidates (CAND)
- Contacts (CTCT)
- Employees (EMPL)
- Requisitions (REQU)

These entities in Taleo Business Edition allow for logging of contact log instances under a contact log table. This table is available through the API as well for programmatic access.

- For retrieving all contact log instances within an associated entity record, use the following URLs:
- **<<HOST_URL>>**/object/contactlog/{id}

For retrieving all contact logs within an associated entity record, use the following URLs:

- **<<HOST_URL>>**/object/**account**/{id}/contactlog
- **<<HOST_URL>>**/object/**candidate**/{id}/ contactlog
- **<<HOST_URL>>**/object/**contact**/{id}/ contactlog
- **<<HOST_URL>>**/object/**employee**/{id}/ contactlog
- **<<HOST_URL>>**/object/**requisition**/{id}/ contactlog

These URLs will also be available as relationship URLs in the response body of entities of these types.

For retrieving all contact log object details, use the following URL:

- <<HOST_URL>>/object/info/contactlog

For field definition, use the following URL (please note that custom and standard URL will output the same fields as this is an enclosed table):

- **<<HOST_URL>>**/object/**contactlog**/description/standard
- **<<HOST_URL>>**/object/**contactlog**/description/custom

The JSON hyperschema field definitions for interviews are available through the following discovery call):

- **<<HOST_URL>>/object/contactlog/**hyperschema.json

Field Name (GUI)	Field Type	Get	Get All	Search	Create / Upsert	Update	Delete
Contact Log		✔	×	×	✔	✔	✔
Id	Integer	Y			N	N	Y (Delete record by ID)
UserId	Table Column	Y			**Y***	Y	
Type	Integer[1]	Y			**Y***	Y	
entityType	Text [2]	Y			**Y***	N	
entityId	Integer	Y			**Y***	N	
entityId2	Integer	Y			N	Y	
subject	Text	Y			**Y***	Y	
comments	Text	Y			N	Y	
Date[3]	Date	Y			**Y***	Y	
creationDate	Date	Y			N	N	
lastUpdated	Date	Y			N	N	

Y* = REQUIRED

Sample Code Contact Log:

Method	Action	Sample Request
GET	Get contact log by ID	…/object/contactlog/69
GET	Get only certain fields of a Contact Log by ID	…/object/contactlog/69?fields=subject|date
DELETE	Delete contact log by ID	…/object/contactlog/69

POST	Create contact log	…/object/contactlog
		
		{
		“contactlog”: {
		“type”: 1,
		“entityType”: “CAND”,
		“entityId”: 72,
		“subject”: “Summary of contact”,
		“comment”:”Details of the contact log instance”,
		“date”: “2008-12-02”
		}
		}
		
		OR for XML
		
		<root>
		<contactlog >
		<type>2</type>
		<entityType>CAND</entityType>
		<subject>Summary of contact</subject>
		<comment>Details of the contact log instance</comment>
		<date>2008-12-02</date>
		</contactlog>
		</root>

POST	Upsert Contact Log entry	…/object/contactlog?upsert=true
		
		{
		“contactlog”: {
		“id”:77,
		“type”: 1,
		“entityType”: “CAND”,
		“entityId”: 72,
		“subject”: “Summary of contact”,
		“comment”:”Details of the contact log instance”,
		“date”: “2008-12-02”
		}
		}
		
		OR for XML
		
		<root>
		<contactlog >
		<id>77</>
		<type>2</type>
		<entityType>CAND</entityType>
		<subject>Summary of contact</subject>

		<comment>Details of the contact log instance</comment> <date>2008-12-02</date> </contactlog> </root>

PUT	Update Contact Log entry	…/object/contactlog/297
		
		{
		“contactlog”: {
		“type”: 1,
		“entityType”: “CAND”,
		“entityId”: 72,
		“subject”: “Summary of contact”,
		“comment”:”Details of the contact log instance”,
		“date”: “2008-12-02”
		}
		}
		
		OR for XML
		
		<root>
		<contactlog >
		<type>2</type>
		<entityType>CAND</entityType>
		<subject>Summary of contact</subject>
		<comment>Details of the contact log instance</comment>
		<date>2008-12-02</date>
		</contactlog>
		</root>

WEB SERVICE REFERENCE: PARSE RESUME FUNCTIONS

Please Note: There is no JSON hyperschema support for resume functions.

The Taleo Business Edition API includes the ability to POST resumes in their original format and receive parsed HR-XML as a response, or include creating candidate records. To do either of these functions, use the following URLs:

Parse Resume for HR-XML resume output:

<<HOST_URL>>/object/candidate/parseresume

Parse Resume for HR-XML resume output and create candidate on the fly with results:

<<HOST_URL>>/object/candidate/resumetocandidate

APPENDIX A—ATTACHMENTS

The TBE REST API currently supports attachments for the following entities:

pmreview
user
employee
account
activity
requisition
candidate
contact
candidate application

Get the list of attachment for a specific entity

GET /{entityType}/{entityId}/attachment

Example:

2. JSON Request URL

http method GET

https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment

JSON response:


```
{
 "response": {
  "attachments": [
   {
    "attachment": {
     "id": 40,
     "entityType": "CAND",
     "entityId": 82,
     "attachmentType": "User_Attachment_Type",
     "description": "About Stacks.pdf",
     "contentType": "application/pdf",
     "userId": 52,
     "attachmentFile": "atchm_40.pdf",
     "creationDate": "2012-06-29",
     "lastUpdated": "2012-06-29",
     "url": "https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment/40",
     "downloadUrl": "https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment/download"
    }
   },
   {
    "attachment": {
     "id": 39,
     "entityType": "CAND",
     "entityId": 82,
     "attachmentType": "User_Attachment_Type",
     "description": "Field Names",
     "contentType": "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
     "userId": 52,
     "attachmentFile": "atchm_39.xlsx",
     "creationDate": "2012-06-29",
     "lastUpdated": "2012-06-29",
     "url": "https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment/39",
     "downloadUrl": "https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment/download"
    }
   }
  ]
 },
 "status": {
```

```
  “detail”: {},
  “success”: true
}
```


3. XML request URL

https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment.xml

XML response:


```
<root>
 <response>
  <attachments>
   <item>
    <attachment>
    <id>40</id>
    <entityType>CAND</entityType>
    <entityId>82</entityId>
    <attachmentType>User_Attachment_Type</attachmentType>
    <description>About Stacks.pdf</description>
    <contentType>application/pdf</contentType>
    <userId>52</userId>
    <attachmentFile>atchm_40.pdf</attachmentFile>
    <creationDate>2012-06-29</creationDate>
    <lastUpdated>2012-06-29</lastUpdated>
     <url>
     https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment/40.xml
     </url>
     <downloadUrl>
     https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment/download.xml
     </downloadUrl>
    </attachment>
   </item>
   <item>
    <attachment>
    <id>39</id>
    <entityType>CAND</entityType>
    <entityId>82</entityId>
    <attachmentType>User_Attachment_Type</attachmentType>
    <description>Field Names</description>
     <contentType>
     application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
     </contentType>
    <userId>52</userId>
    <attachmentFile>atchm_39.xlsx</attachmentFile>
    <creationDate>2012-06-29</creationDate>
    <lastUpdated>2012-06-29</lastUpdated>
     <url>
     https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment/39.xml
     </url>
     <downloadUrl>
     https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment/download.xml
     </downloadUrl>
    </attachment>
   </item>
  </attachments>
 </response>
 <status>
```

```
<detail/>
<success>true</success>
</status>
</root>
```

Attachment Type(s)

The Attachment Type field can be used to differentiate between Candidate uploaded attachments and User uploaded attachments.

Values:

- `<attachmentType>Candidate_Attachment_Type</attachmentType>`
- `<attachmentType>User_Attachment_Type</attachmentType>`

Get the a specific attachment entity by Id

GET /{entityType}/{entityId}/attachment/{id}

Example:

http method GET

https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment/39

or https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment/39.xml

Response:


```
{
 "response": {
  "attachment": {
   "id": 39,
   "entityType": "CAND",
   "entityId": 82,
   "attachmentType": "User_Attachment_Type",
   "description": "Field Names",
   "contentType": "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
   "userId": 52,
   "attachmentFile": "atchm_39.xlsx",
   "creationDate": "2012-06-29",
   "lastUpdated": "2012-06-29",
   "url": "https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment/39",
   "downloadUrl": "https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment/39/download"
  }
 },
 "status": {
  "detail": {},
  "success": true
 }
}
```


Create an attachment

POST /{entityType}/{entityId}/attachment

Using "multipart/form-data" POST request.

Optional request parameter: description

If the request parameter description is not sent, the attachment entity description field will use the file name as the description.

Example:

http method POST

https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment

Header must include: {:file=>#<File:path/create_file.doc>,:multipart=true}
For additional information about multipart/form-data, refer to mutilpart/form-data specification

(www.w3.org). http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4.2

Example header for posting a Word attachment or resume:

{:file => File.new("/local/path/zip/resume.doc", 'rb'), :content_type =>
"application/msword", :multipart => true}, {:cookies => {:authToken => "webapi26117672355175654454"}})

Update an attachment

POST /{entityType}/{entityId}/attachment/{id}

The update will replace the old attachment with the new attachment and description.

Using "multipart/form-data" POST request.

Optional request parameter: description

If the request parameter description is not sent, the attachment entity description field will use the file name as the description.

Example:

http method POST

https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment/{id}

Header must include: {:file=>#<File:*path*/create_file.doc>,:multipart=true}

For additional information about multipart/form-data, refer to mutilpart/form-data specification

(www.w3.org). http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4.2

Example header for posting a Word attachment or resume:

{:file => File.new("/local/path/zip/resume.doc", 'rb'), :content_type =>
"application/msword", :multipart => true}, {:cookies => {:authToken => "webapi26117672355175654454"}})

Delete an attachment

DELETE /{entityType}/{entityId}/attachment/{id}

Example:

http method DELETE

https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment/{id}

Download an attachment

GET /{entityType}/{entityId}/attachment/{id}/download

This API will download the attachment binary file. The response header will include header attributes such as Content-Type for the file mime type etc.

Example:

http method GET

https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/attachment/{id}/download

RESUME Attachment

The TBE REST API supports the candidate resume upload, parsing and download.

Download an attachment

GET candidate/{id}/resume

This API will download the attachment binary file. The response header will include header attributes such as Content-Type for the file mime type etc.

Example:

http method GET

https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/resume

Update a candidate resume

POST /candidate/{id}/resume

The update will replace the old attachment with the new attachment and description.

Using "multipart/form-data" POST request.

Optional request parameter: description

If the request parameter description is not sent, the attachment entity description field will use the file name as the description.

Example:

http method POST

https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/82/resume

Header must include: {:file=>#<File:path/create_file.doc>,:multipart=true}
For additional information about multipart/form-data, refer to mutilpart/form-data specification

(www.w3.org). http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4.2

Example header for posting a Word attachment or resume:

{:file => File.new("/local/path/zip/resume.doc", 'rb'), :content_type =>
"application/msword", :multipart => true}, {:cookies => {:authToken => "webapi26117672355175654454"}})

Parse a candidate resume

POST /candidate/parseresume.xml

This API will parse the uploaded candidate resume document and return the resume in a XML format.

Using "multipart/form-data" POST request.

Example:

http method POST

https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/parseresume.xml

Header must include: {:file=>#<File:path/create_file.doc>,:multipart=true}

For additional information about multipart/form-data, refer to mutilpart/form-data specification (www.w3.org). http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4.2

Example header for posting a Word attachment or resume:

{:file => File.new("/local/path/zip/resume.doc", 'rb'), :content_type =>
"application/msword", :multipart => true}, {:cookies => {:authToken => "webapi26117672355175654454"}})

Update a candidate resume

POST /candidate/resumetocandidate

This API will extract candidate information from the uploaded candidate resume attachment, and create a new resume.

Using "multipart/form-data" POST request.

Optional request parameter: referredBy

If the request parameter description is not sent, the attachment entity description field will use the file name as the description.

Example:

http method POST

https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidate/resumetocandidate

Header must include: {:file=>#<File:path/create_file.doc>,:multipart=true}

For additional information about multipart/form-data, refer to mutilpart/form-data specification (www.w3.org). http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4.2

Example header for posting a Word attachment or resume:

{:file => File.new("/local/path/zip/resume.doc", 'rb'), :content_type => "application/msword", :multipart => true}, {:cookies => {:authToken => "webapi26117672355175654454"}})

Bulk Upload candidate resumes

POST /candidate/bulkresumeimport

This API will asyncrously extract the files contained in a .zip file and create candidates with contact information based on the content of each resume. The resume will also be attached to the created candidate. File types supported for the resumes in the .zip file are the same as manual upload within TBE application .txt., .doc, pdf, etc.

Additional parsing of the resume content will be done according to the setting on the user used to execute the API request. Parsing options for the user can be viewed and set in TBE in the User's settings. My Settings=>Resume Parsing Settings

Supported Parameters:

The Bulk Resume Upload API supports the following parameters, all optional.

- requisitionId – The requisition to which the Candiate should be attached.
- sourceId – the Source to set on the Candidate and Candidate Application if applicable.
- StatusId – the Status that should be assigned to the Candidate upon creation.
- userId – the TBE user who should be notified of the results of the Bulk Upload if different than the user executing the request.

Example:

- ../object/candidate/bulkresumeimport?requisitionId=99 &sourceId=9&statusId=88& userId=99

Response:

This API is asycronous. The results of the candidate creation for the resumes contained will be emailed to the TBE User executing the API Request or the user specified in the userId parameter.

If the initial request is successful and the .zip file is successfully submitted, the API Response will contain:

- Response URL: '…/v1/object/candidate/search'
- Status = success: true

If the initial request is not valid, the API response will contain:

- Status = success: false

- Error text.
    - File not found.
    - File does not contain at least one file.

The Bulk Resume Upload uses “multipart/form-data” POST

request.

Example:

http method POST

https://stgweb1.tbetaleo.com/QANA3/ats/api/v1/object/candidat

e/resumetocandidate

Header must include: {:file => “path/name”, :content_type => “application/zip”, :multipart => true}

For additional information about multipart/form-data, refer to mutilpart/form-data specification

(www.w3.org). http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4.2

Example header for posting a Word attachment or resume:

{:file => File.new(“/local/path/zip/resume.doc”, ‘rb’), :content_type => “application/msword”, :multipart => true}, {:cookies => {:authToken =>“webapi26117672355175654454”}})

Document History

Date	Changed By	Comments	Version
		NOTE: Previous changes have been removed from this table	
11/20/2019	Kathy DiPaola	Updates: • Authentication sections on pages 8,13 and 24 to reference cookie based authentication. • URL structure on page 11 • Post methods on page 25	19D
08/14/2020	Ravindra Bisht	Updates: • Added Candidate Answer information on page 21 and pages 91-95	20C