API Reference

Integrate Upwork functionalities to your web-based or mobile apps

Upwork Developers Site offers you access to our web services
to **build** your own applications and to **integrate** our features and workflow
to your dashboards, websites and management systems.
Please read Terms of use prior to using the Upwork Public API.

Featured functionalities

- Authenticate and authorize your users with Upwork
- Search for freelancers and find jobs
- Create job postings
- Invite freelancers to an interview
- Make job offers
- Manage existing engagements
- Make custom payments
- Close contracts
- Manage activities for your team
- Manage, send and receive messages

Libraries

Here you can find a list of available libraries for several languages.

API Entry Point

`https://www.upwork.com/api`

For Reports resources:

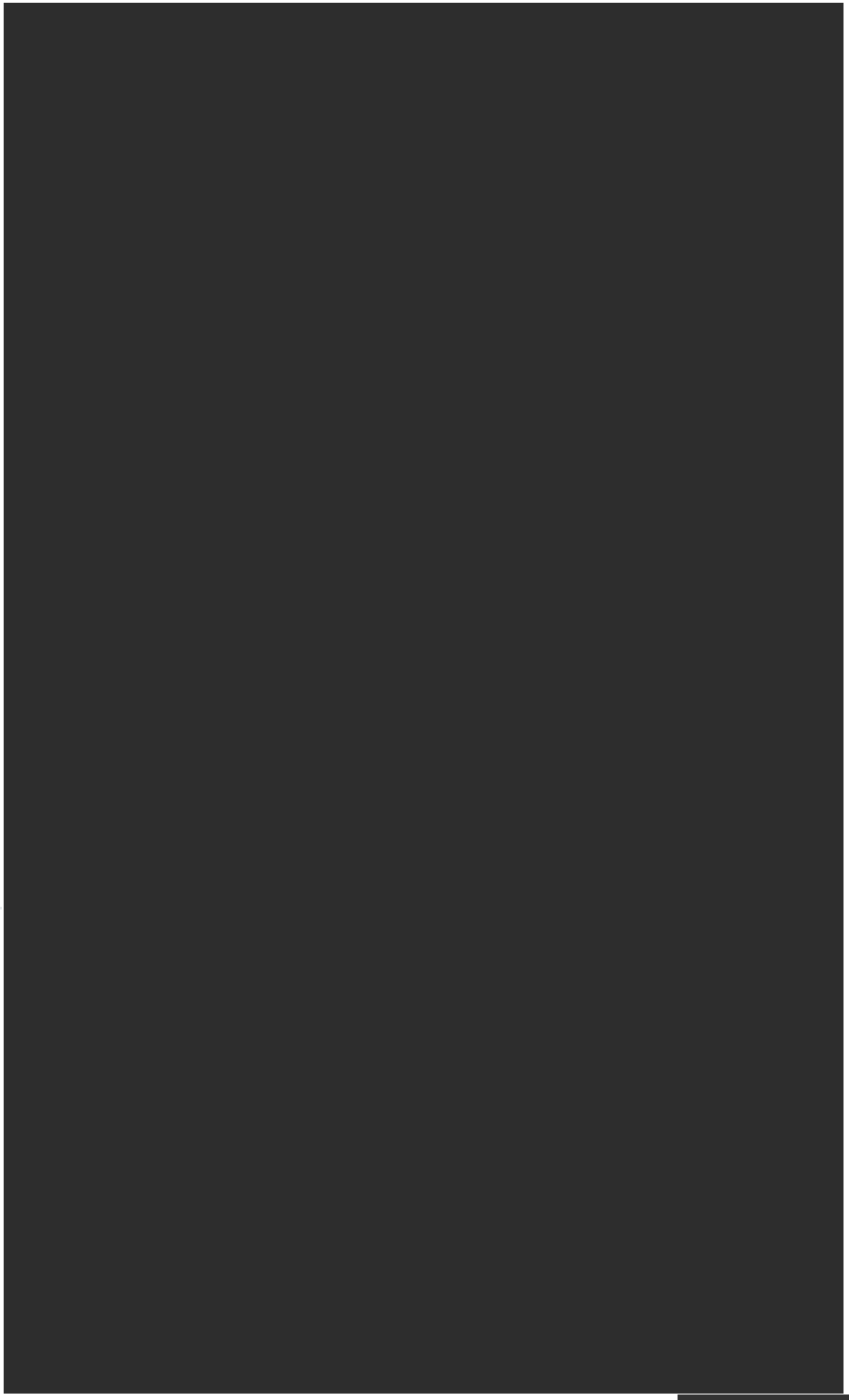
`https://www.upwork.com/gds`

• Retrieve time and financial reports for freelancers, teams and companies
• Manage work diary snapshots
• Retrieve metadata information - available categories, tests, skills, regions, etc.

Getting started is easy. Check out our Getting Started section to learn about the core concepts of our API, such as supported authentication methods, languages and formats.

This page contains complete API reference documentation about our API technical implementation, just use your browser search or navigation panel on the left.

Have a question? Stop by the FAQs and Forum, or contact our Support team. Don't forget to visit our Recent changes section to check out the new releases and features.



Getting Started

Preparation

Before you start using Upwork API, you need to register your application and obtain your client credentials (a.k.a consumer key and consumer secret). To register, you just fill in basic information about your application and its

intended use.

Register a new application

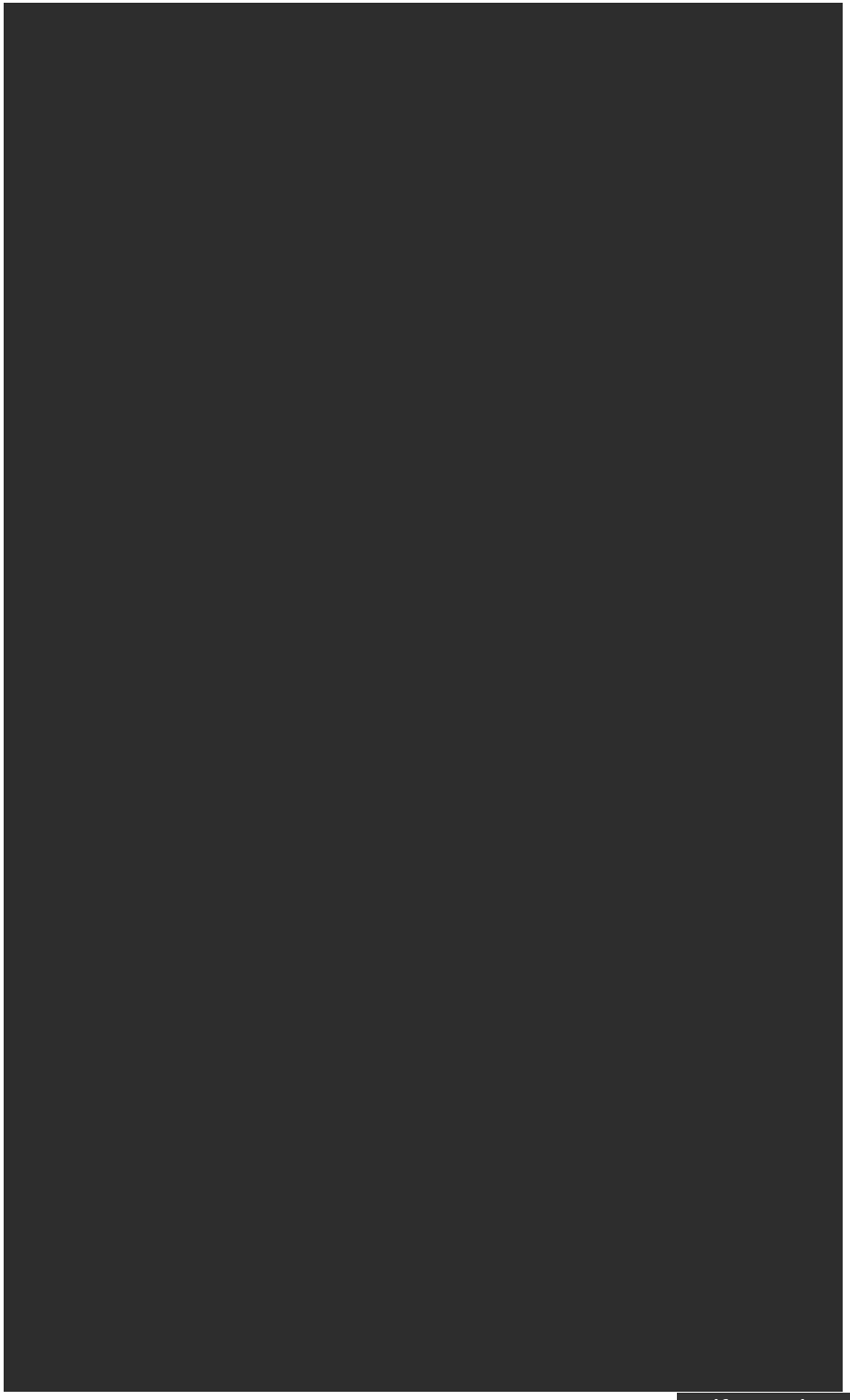
See the list of your registered applications (keys)

Note

Every new request for an API Key will be reviewed. For a quick positive decision, you need to comply the following conditions:

1. your Upwork profile must have:
   a. a valid name in your profile (no fake names, no company names)
   b. a full valid address (including apartment number) - either in your personal or your company's profile
   c. a valid profile portrait

2. provide a clear description on what you are planning using our APIs for (this info is required when you apply for a key). Do not forget add a note if you are going to use your application internally or publicly, or for any other purposes.
3. agree to use the Upwork API in a manner that does not exceed a reasonable request volume. Our daily allowed limit is 40K reqs.
4. refrain from using the Upwork logo, brand name, brand theme/colors and trademarked content in your application



Application Permissions

When you register a new application, you are asked to define a list of permissions to access third-party resource data. You can later modify the permissions you request, but note that doing so makes all previously generated access tokens invalid. You will also have to request authorization from resource owners again.

Authentication

To access Upwork API, you need to go through an authentication process. Currently, we support the OAuth 1.0 method.

Note

You need to authenticate for all requests following the OAuth 1.0, RFC 5849.

The authentication process is simple. See the authentication section for a detailed description.

Libraries and Tools

A library is a collection of behavior implementations, written in a specific programming language. Libraries help you

in a specific programming language. Libraries help you make system calls without the need to re-write code over and over again.

These are the libraries built specifically to support Upwork API:

- Python
- PHP
- Ruby
- Java (+ Android, Maven, etc)
- Javascript/NodeJS
- Perl
- GO

On the right-hand side you can see the examples of how to make API calls with the chosen library.

Additionally, you can find a list of libraries for the language of your choice to help you with the OAuth mechanism here. To start using most of these libraries, you just need to provide your client credentials and the necessary URLs.

Architecture style

Upwork follows the REST style. Thus, our API resources are accessed by explicitly using HTTP methods (`GET`, `POST`, `PUT` and `DELETE`) and following the protocol as defined by

```
$ mkdir myapp
$ cd myapp
$ npm install upwork-api
$ cp node_modules/upwork-
api/example/example.js example.js
```

Edit the `example.js` *file to include your oAuth keys and run the application:*

```
$ node example.js
```

RFC 3986 standard.

Note

`PUT` and `DELETE` methods have couple important constraints:

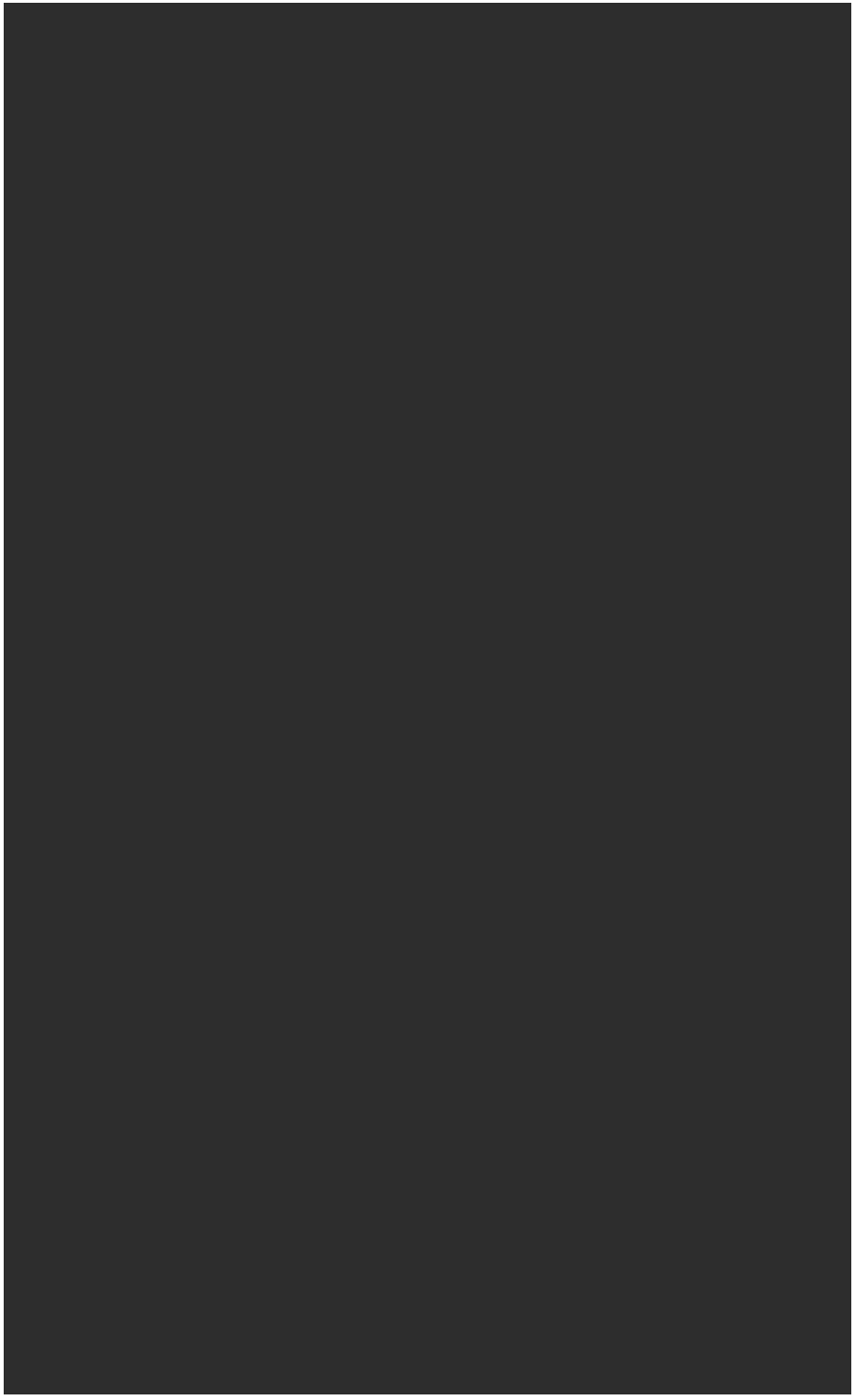
- Methods are available only for JSON and XML requests.
- `Content-Type` (**RFC 1049**) header must be equal to `application/json` (**RFC 4627**) for a JSON request.
- `Content-Type` (**RFC 1049**) header must be equal to `application/xml` (**RFC 2376**) for an XML request .
- All parameters must be encoded in JSON format `{"key1": "value1", "key2": "value2"}` or XML format `<request><key1>value1</key1></request>` respectively.
- All parameters data must be sent as raw post data encoded as described above .

Request structure

The base URLs for all Upwork API requests are:

- https://www.upwork.com/api/ - main entry point
- https://www.upwork.com/gds/ - an entry point for Report resources over GDS protocol

Note



All API requests must be made over HTTPS. API requests made over plain HTTP will be redirected to their equivalent HTTPS.

Cross-domain requests

Getting domain data from a domain different from the REST servers can cause failures. This is why we standardized our API for cross-domain requests. This makes it possible for you to use standard jQuery functions with the callback parameter.

Note

You must add `&callback=?` and `oauth_xxx` parameters to all your cross-domain requests, otherwise the API will not return standardized jQuery data. You can supply a `?callback` parameter to any request in order to enable JSON-P wrapping. This is useful for cross-domain AJAX requests.

You can use this code to make JSON requests using jQuery:

```
$(document).ready(function(){

$.getJSON("https://www.upwork.com/api/pro
files/v2/search/jobs.json?
q=java&callback=?&oauth_params=xxxxx",
       function(response){
          alert('Server Time: ' +
response.server_time);
       });
});
```

You can also use the code below to make JSONP request with jQuery:

```
  $.ajax({

url:"https://www.upwork.com/api/profiles/
v2/search/jobs.json?q=java&callback=?
&oauth_params=xxxxx",
     dataType: 'JSONP',
     success:function(json){
         alert("Success:
"+json.server_time);
     },
     error:function(){
         alert("Error");
```

Versioning

We version our API to support forward and backward compatibility. This means that new implementations don't affect applications which depend on our API. The API version you target is defined the first time you make an API request. Every time we make backward-incompatible changes to the API, we release a new version. To avoid affecting your applications, we don't modify your version until you're ready to upgrade. Note that our versioning process is made at the resource level.

Our API supports three ways of requesting a version of the API (see options on the right-hand side).

Note

We also support locating the version number on the second sub-path (if needed due to framework requirements). For example: `/api/v1/auth/info.json`

Response format

Our API supports both XML and JSON response formats. We suggest you append the format specifier (`.json` or

```
    },
});
```

Three ways of specifying a version for the API call:

standard *(recommended)*
the version is located on the ***third*** *subpath in the URL:*

```
/api/auth/v1/info.json
```

via ``X-Upwork-Version`` header *(has higher priority):*

```
X-Upwork-Version: v1
```

via ``Accept`` header *(has lower priority)*

```
Accept: application/vnd.upwork.api-v1+json
```

`.xml`) to the request URI to explicitly select JSON or XML. Otherwise, the API may default to either format. You can also use the `Accept` request-header, passing `application/xml` (RFC3023) or `application/json` (RFC4627) to define the response format.

Response elements

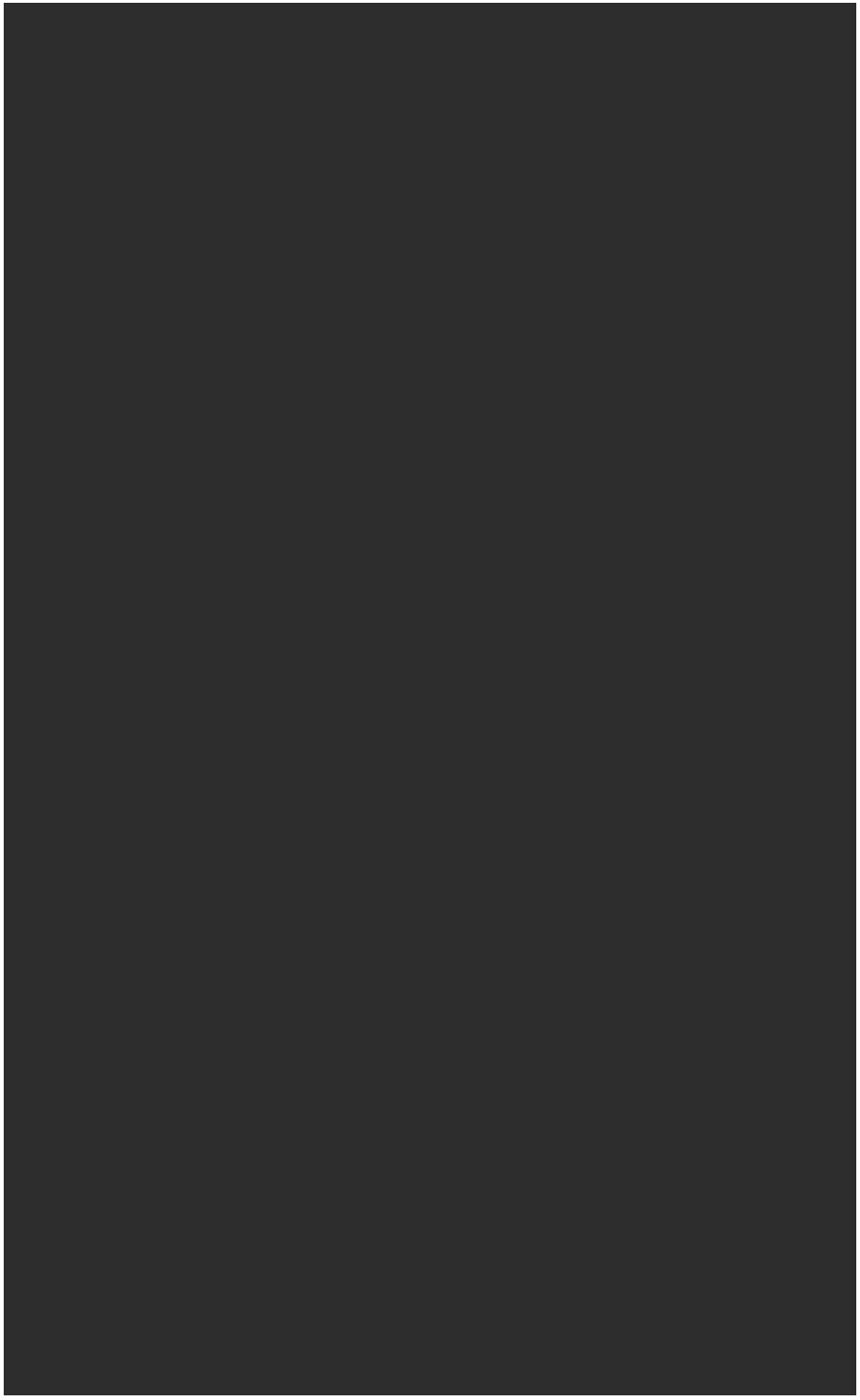
All responses include at least two elements:

`auth_user` - Information about the user making the request
`server_time` - Server time, in UNIX timestamp format

Response properties are always one of the following data types:

integer:	A positive or negative whole number
string:	Unicode text
URL:	A string that conforms to RFC 1738
timestamp:	A date expressed as seconds elapsed since January 1st, 1970

Pagination



Many calls support pagination. Although details of pagination and ordering may differ (see specific information for each resource in the API reference), API calls that do support pagination mostly do so by using the `page` and `paging` parameters. The format of these parameters is: `$offset;$count`, for example: `page=20;10`

The default values are `0;10`. Count can be restricted to <= *N*, where *N* is the maximum page size, found in the parameter description.

Encoding

We use UTF-8 encoding. UTF-8 encodes each Unicode character as a variable number of 1 to 4 octets, where the number of octets depends on the integer value assigned to the Unicode character. It is an efficient encoding of Unicode documents that use mostly US-ASCII characters because it represents each character in the range U+0000 through U+007F as a single octet.

UTF-8 is the default encoding for XML and since 2010, it has become the dominant character set on the Web.

Rate Limits

Rate limiting of the API is primarily considered on a per-IP

basis. We allow you to make 40 requests per window per IP. We use 1min windows. When an application exceeds the rate limit for our APIs, we will return an HTTP 429 “Too Many Requests” response code. If your application gets rate limited, then it will be unblocked within a minute since the last request that got rate limited.

To avoid being rate limited, please use caching in your application. Store API responses in your application if you expect a lot of use. For example, don’t try to call the Upwork API on every page load. Instead, call the API infrequently and load the response into a local cache. When users hit your website load the cached version of the results.

Error Handling

Upwork API returns standard HTTP error codes and provides additional error information in the response body (when allowed by HTTP specification), and in the special HTTP headers.

Headers examples:

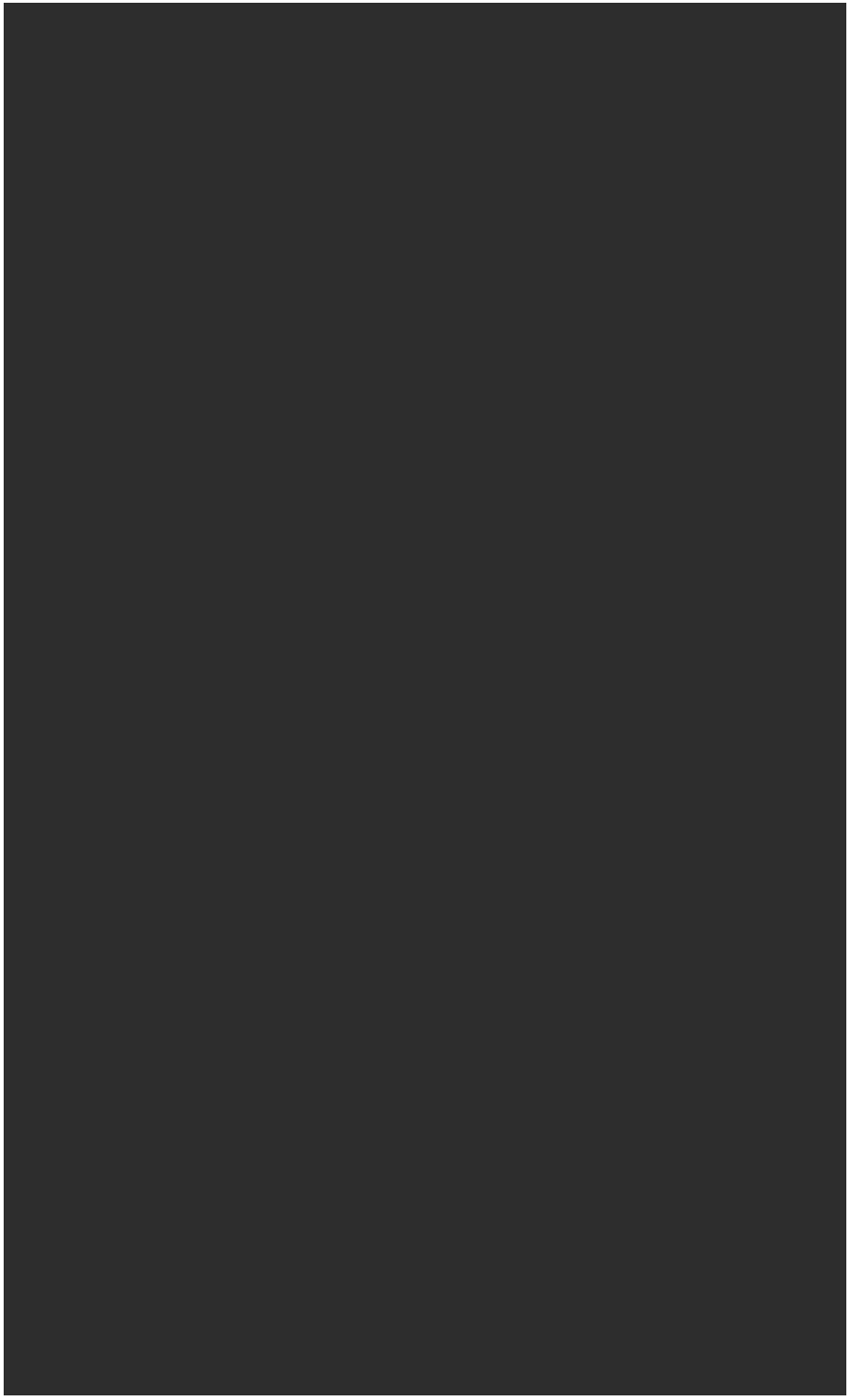
X-Upwork-Error-Code - Internal error code, useful in case you contact our Support team
X-Upwork-Error-Message - Additional error information

Body example:

```
{
    server_time: 1320658441
    error: {
        status: "403" // equal to HTTP
status
        code: "174"   // internal error
code, useful in case you contact Support
team
        message: "Insufficient permission
to list offers" // additional error
message
    }
}
```

The following table describes the most common HTTP error messages you may receive and possible solutions.

400 - Bad Request:	The request could not be understood by the server due to malformed syntax. • You should not repeat the request without modifications. • Check the error message and response body to obtain information about the possible reason. • Visit our Documentation site to learn more about the requirements, limits and allowed values of the request parameters and fields.
401 - Unauthorized:	The request requires user authentication. • Check the error message and response body to obtain information about the possible reason. • Authenticate by following OAuth 1.0 protocol.



 to the top

back to Upwork

Introduction

Getting St…

Preparation

Applicati…

Authentic…

Libraries …

Architect…

Request …

Cross-do…

Versioning

Respons…

Respons…

Pagination

Encoding

Rate Limits

Error Ha…

Terminology

Authentica…

Users

Public Prof…

Jobs

Companie…

• Visit our Authentication page to learn more about Upwork authentication process.

403 - Forbidden: The server understood the request, but is refusing to fulfill it.

• Authorization will not help and the request SHOULD NOT be repeated.
• Check the error message and response body to obtain information about the possible reason.
• Visit our Documentation site to learn more about the requirements, limits and allowed values of the request parameters and fields.

404 - Not Found: The Request-URI did not match any resource in the server.

• Check the error message and response body to obtain information about the possible error cause.
• Visit our Documentation site to

PYTHON PHP RUBY JAVA NODE PERL GO

learn more about the requirements, limits and allowed values of the request parameters and fields.

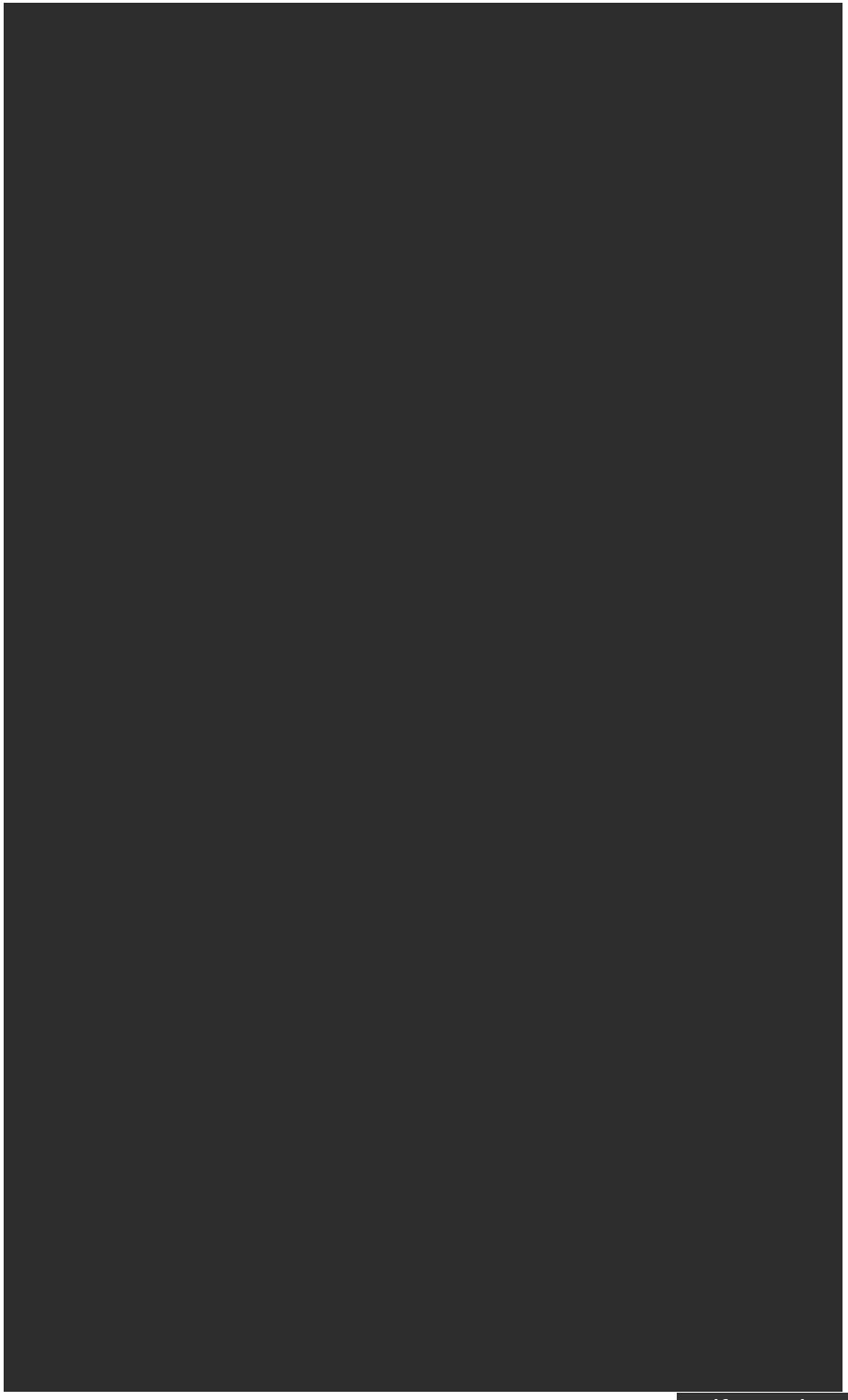
413 - Request Entity Too Large

The server is refusing to process a request because the request entity is larger than the server is able to process.

- Check the error message and response body to obtain information about the possible reason.
- The server MAY close the connection to prevent you from continuing the request. If the condition is temporary, you MAY try again.
- Visit our Documentation site to learn more about the requirements limits and allowed values of the request parameters and fields.

429 - Too Many Requests

The server is refusing to process a request because an application has sent too many requests in a

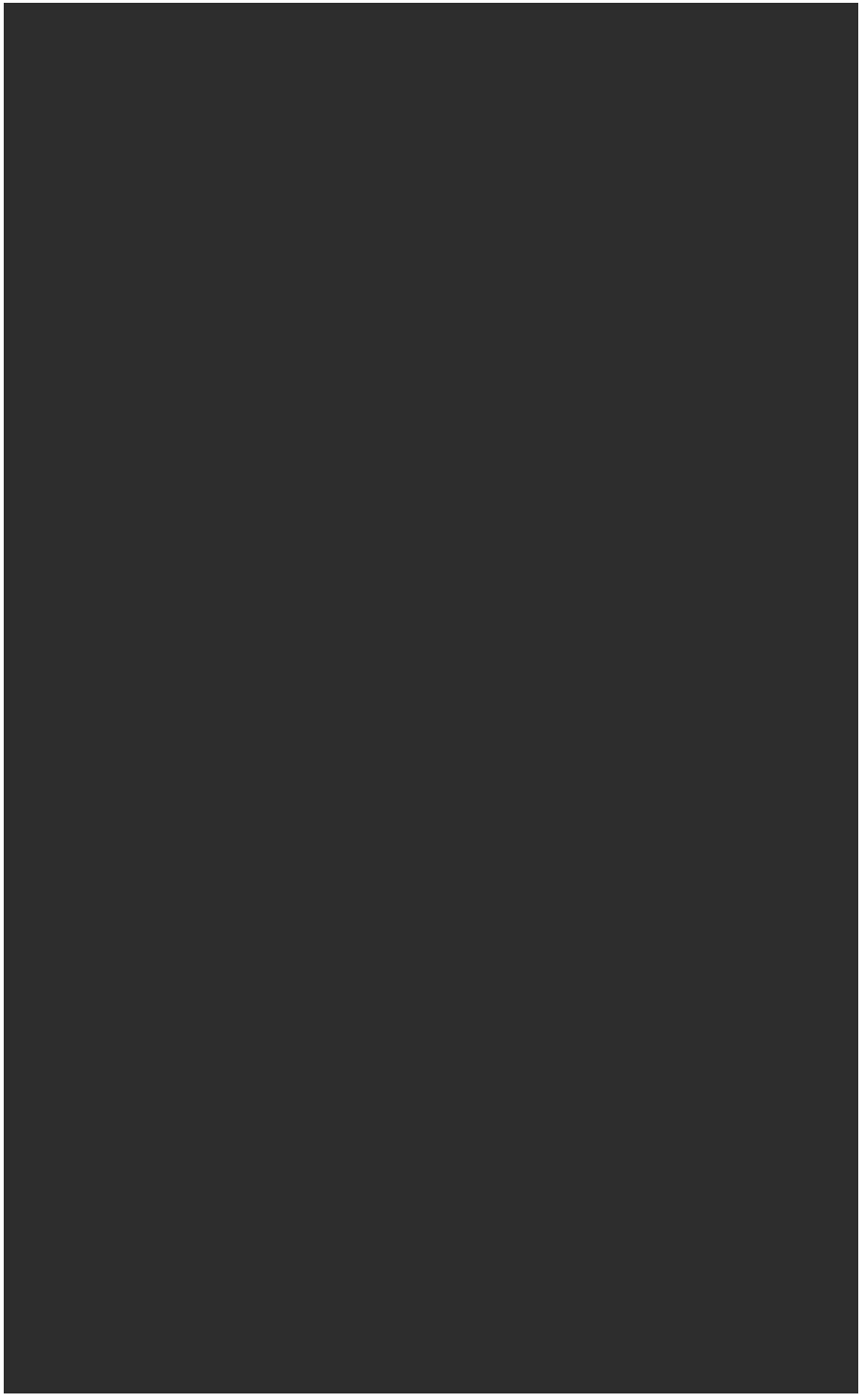


given amount of time. Intended for use with rate limiting schemes.

• You should reduce the number of simultaneous requests.
• You should comply with our rate-limit-policy, i.e. avoid exceeding daily limits.

In case you are still unable to work the error out, contact our Support team. They are more than happy to help you further. Please, include the following information on your message:

• error code
• error message
• public key of your (web server) application
• Upwork username used in requests
• full URL, please include all parameters (keys, signature, token) and headers used
• time of request (if possible)
• method of the request (`GET`/`POST`/`PUT`/`DELETE`. Note that `PUT` and `DELETE` are overloaded via `POST+http_method=<method>`.)
• your environment (OS, library, programming language)
• how did you run the request? Mobile, desktop, console, web (for web please specify your browser), etc.



- response headers and HTTP status
- response body

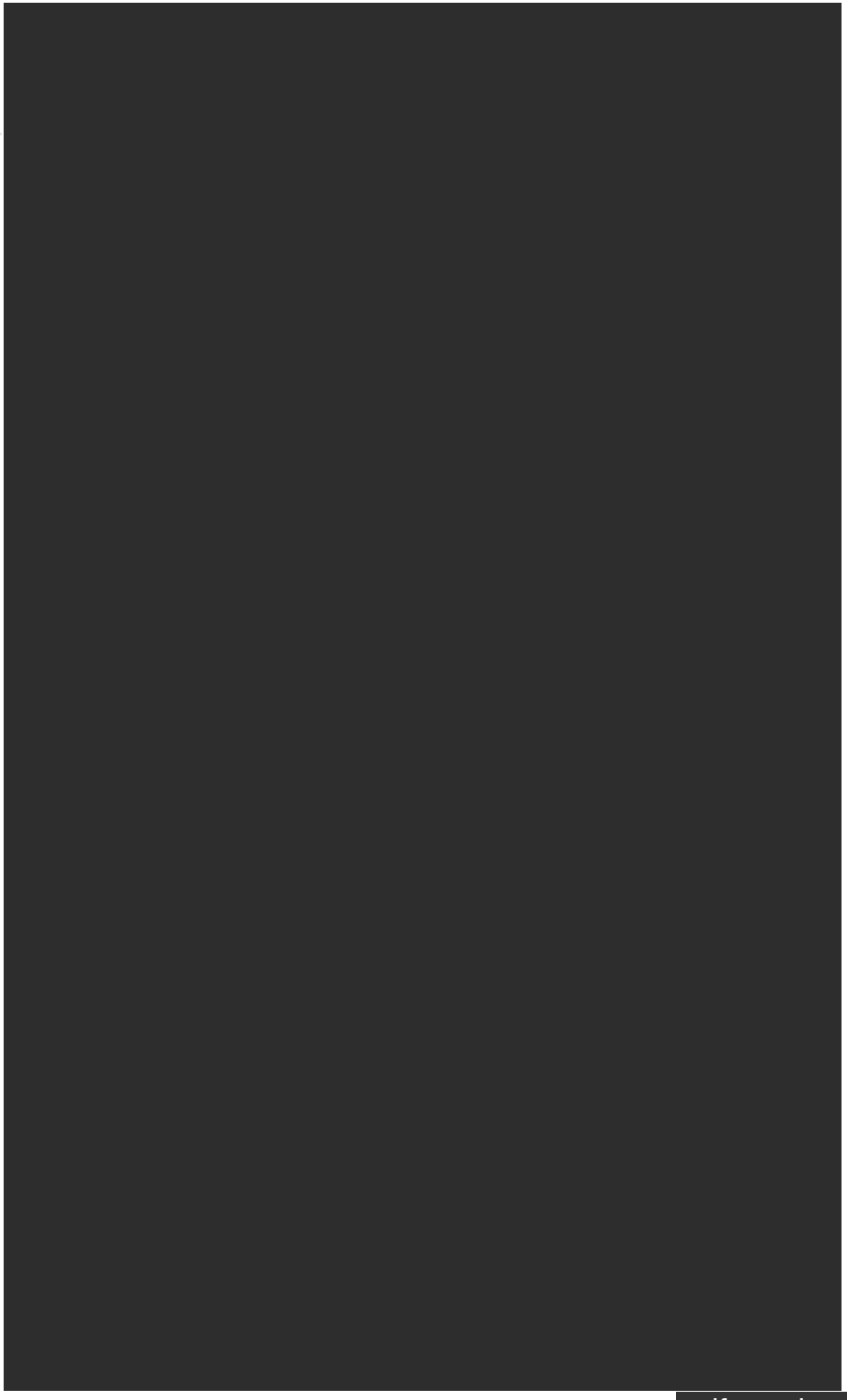
Terminology

In order to use Upwork API, it is important to have a good understanding of the underlying concepts of Upwork Platform.

Users, clients, freelancers

User: An Upwork account holder. A user can have multiple roles simultaneously: client, freelancer, agency staffing manager, and so on. A user must be a single person and account sharing is not allowed. Others can be added to your *company* or *agency*.

Freelancer: A person who can apply to jobs, work, and get paid by clients via Upwork. In API parameters and arguments, sometimes the terms `provider` and `contractor` are used to mean *freelancer*. Those are legacy names for the term.



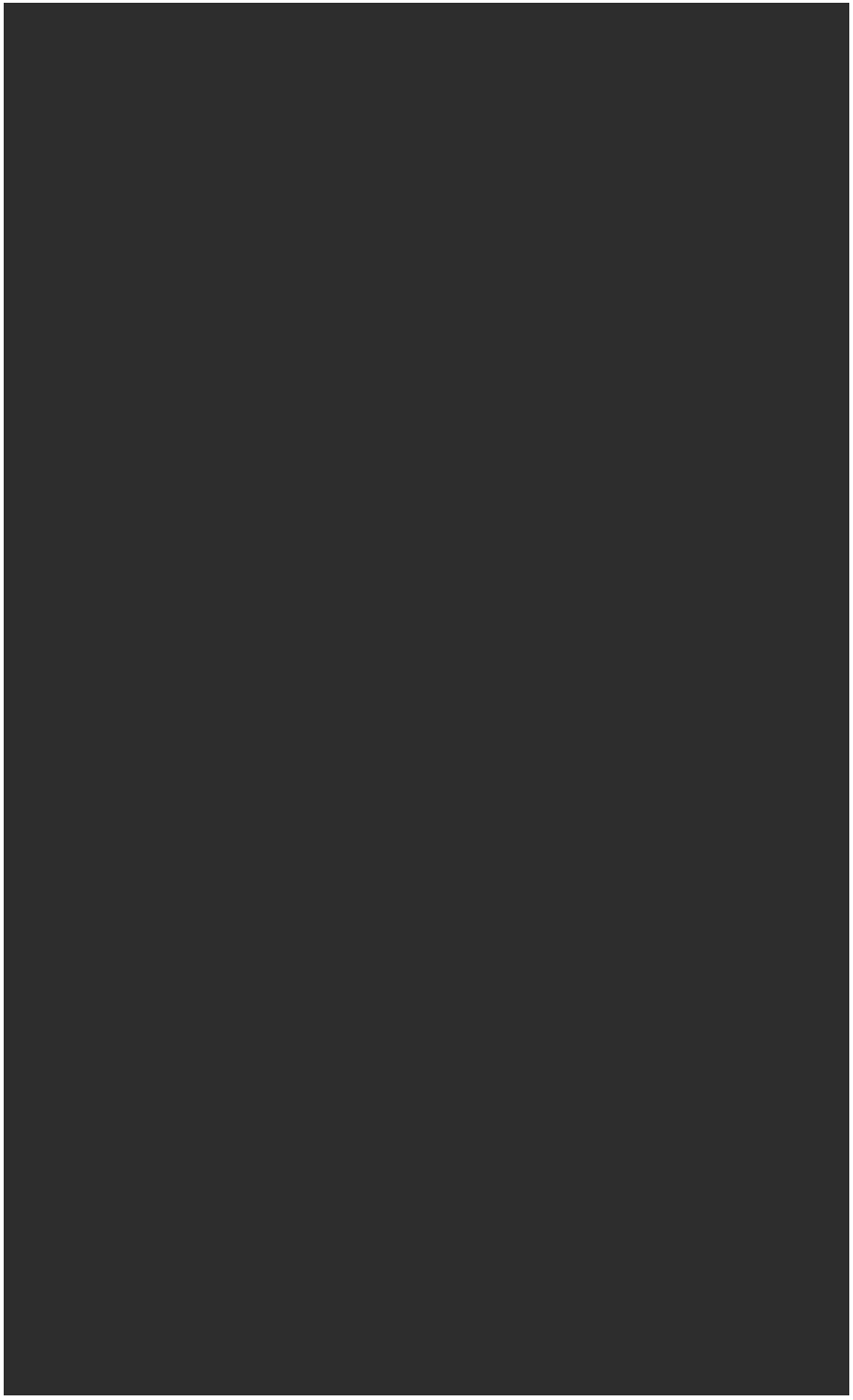
Client: A person who, as part of a company, can hire Upwork freelancers. In API parameters and arguments, sometimes the term `buyer` is used to mean *client*- it's a synonym.

Agency contractor: There are two types of agency contractors. Exclusive agency contractors can only work on behalf of the *agency*. The agency controls all their earnings. They cannot control their rates. Non-exclusive agency contractors can work for multiple agencies and/or independently. An agency can only control their agency jobs. Agency contractors can access worked hours' reports, but not their earnings reports.

Companies, teams, agencies

Company: A primary financial entity and organizational unit. Larger companies can request to be divided into multiple teams.

Team: A basic unit of organization.



Members of the team can see each other and work on common tasks. Most permissions are also relative to the team. At the beginning, each company has one dedicated team room, but others can be added later.

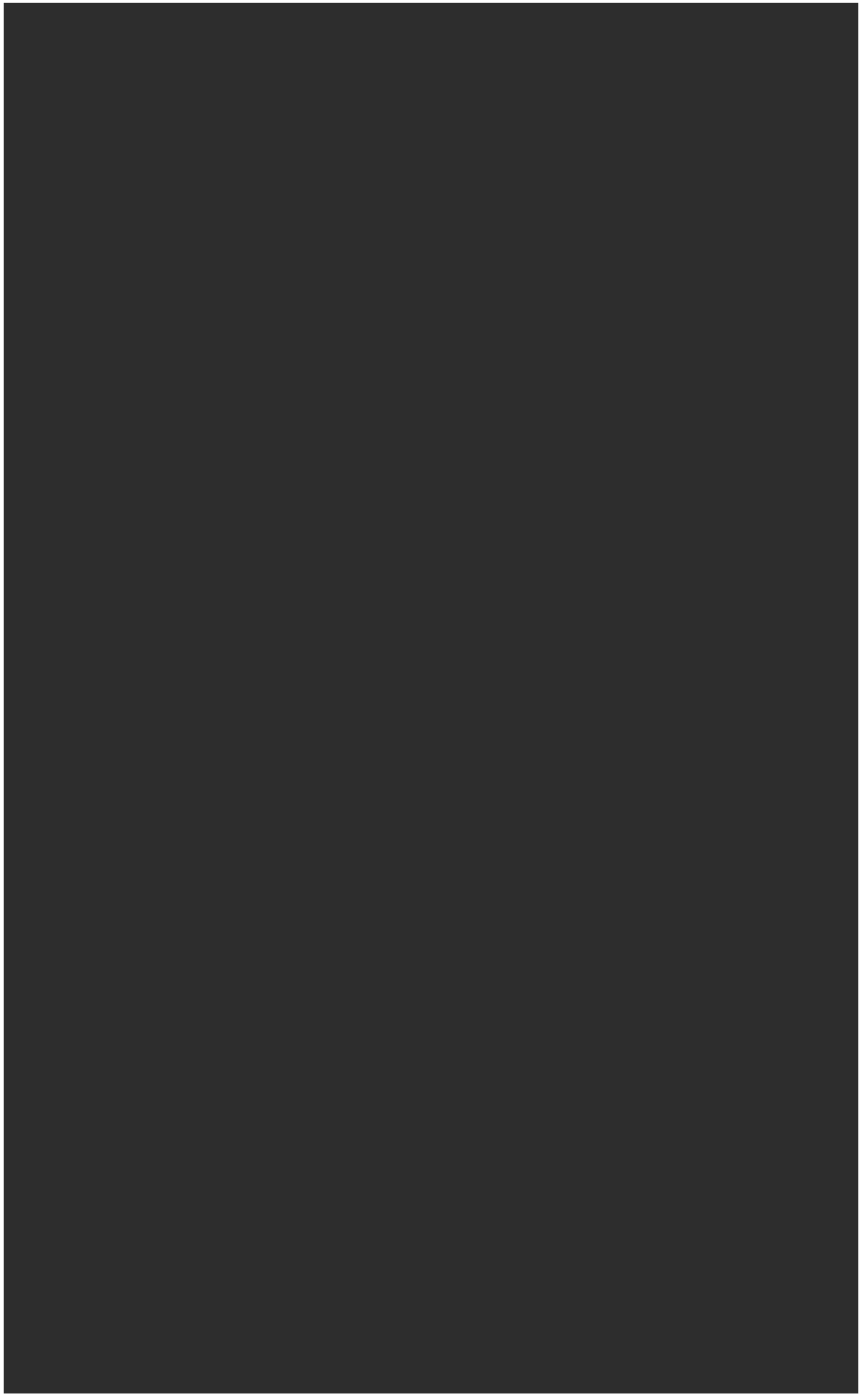
Agency: Acts as intermediary between the *client* and the *freelancer*. The client still hires a specific person, but the contract is paid to the agency. Agency size and staffing manager involvement varies widely.

Job postings, offers and contracts

Job posting: Description of the work to be done and the application guidelines for freelancers.

Offer: Proposal of contract terms as part of the hiring process. It can be initiated by the *client* or the *freelancer*, but it must be agreed upon by both parties.

Contract: The terms and conditions that rule a particular job between a



client and a *freelancer*.

Fixed-price contract: Flat-rate contract or payment for milestones. It is not covered by the Upwork Guarantee.

Hourly contract: Contract billed per hour based on the *Work diary* registry. It is covered by the Upwork Guarantee.

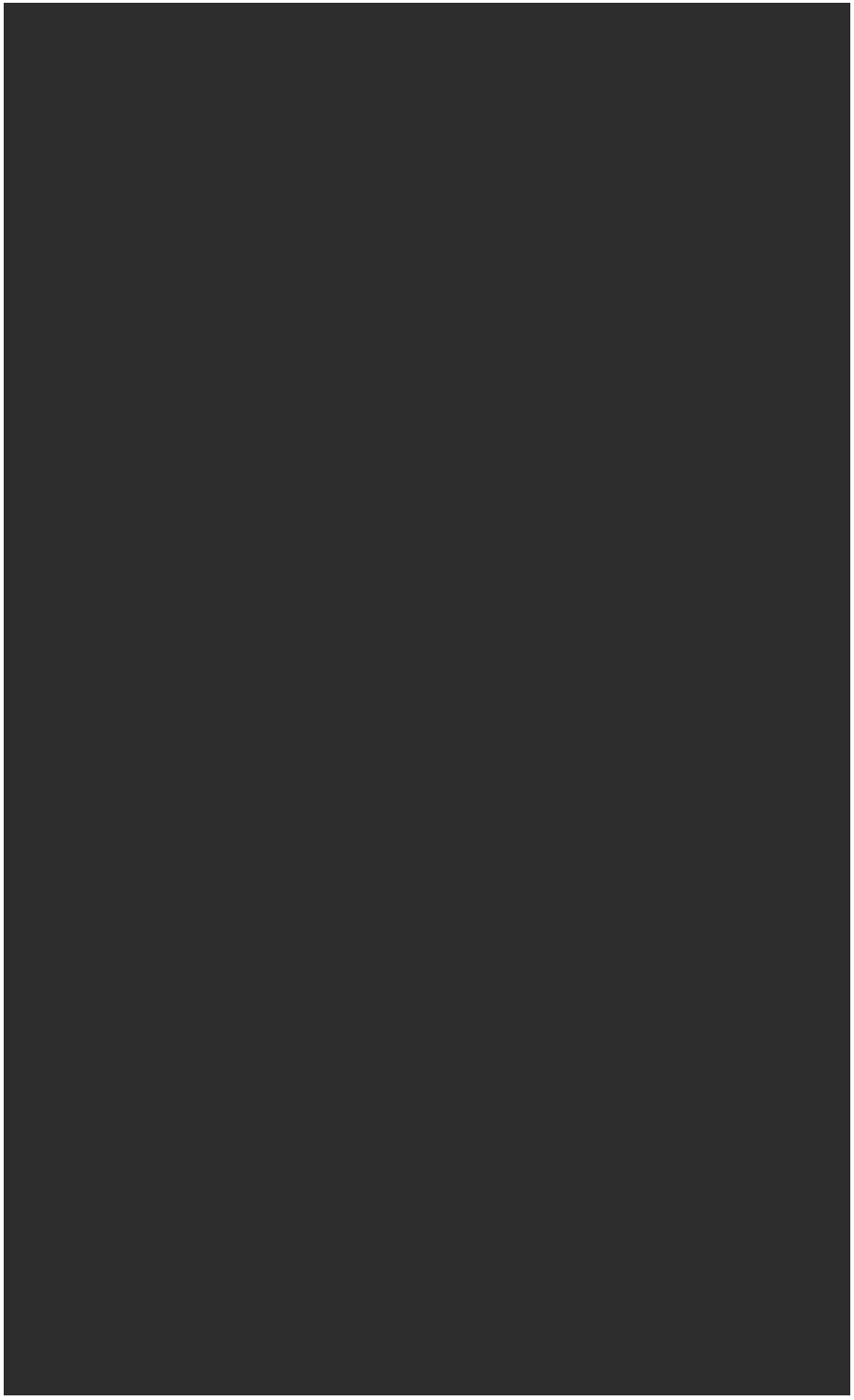
Charge Amount: The amount of money that is deducted from the company's financial account (Upwork fee included).

Amount: The amount of money that a *freelancer* or an *agency* receives as payment (Upwork fee not included).

Work diary: The *Work diary* acts as a visual time card. It is the basis for Upwork's automated weekly billing system. It shows clients hours worked and work-in-progress screenshots.

Roles and permissions

Permissions define what a user can do in the context of a given *team* or a *company*. All permissions are relative to the



team room, except for *owner*, which is relative to the company.

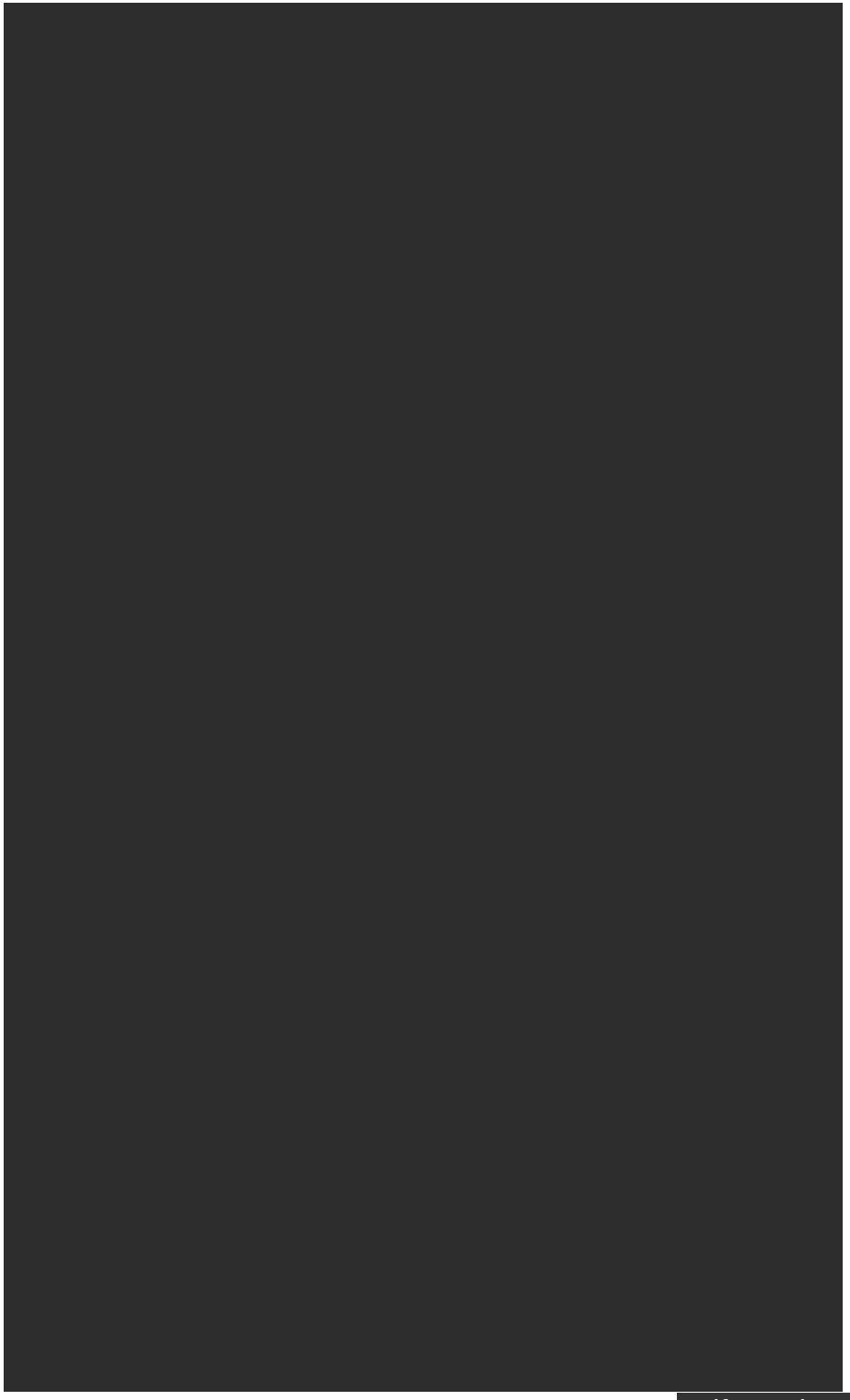
Owner: The lead *admin* responsible for the *company* and all its teams. By default, the company creator is the *owner*, but ownership can be transferred by contacting Customer support. Only the owner can create additional *teams* within a company.

Admin: A user with permissions to: add and remove *team* members, control everyone's permissions, and manage the team's financial account.

Financial access: A user with full access to the team's financial account. This user cannot control team members or permissions.

Hiring manager: A user with full client permissions to post jobs, invite to an interview, make offers, hire, set weekly limits, change rates, make payments, end contracts, and leave feedback.

Recruiter: A user with limited client



	permissions to post jobs, invite, interview, and manage candidates.
Staffing manager:	A user with control over agency contractors' job applications, rates and agency reports.

Authentication

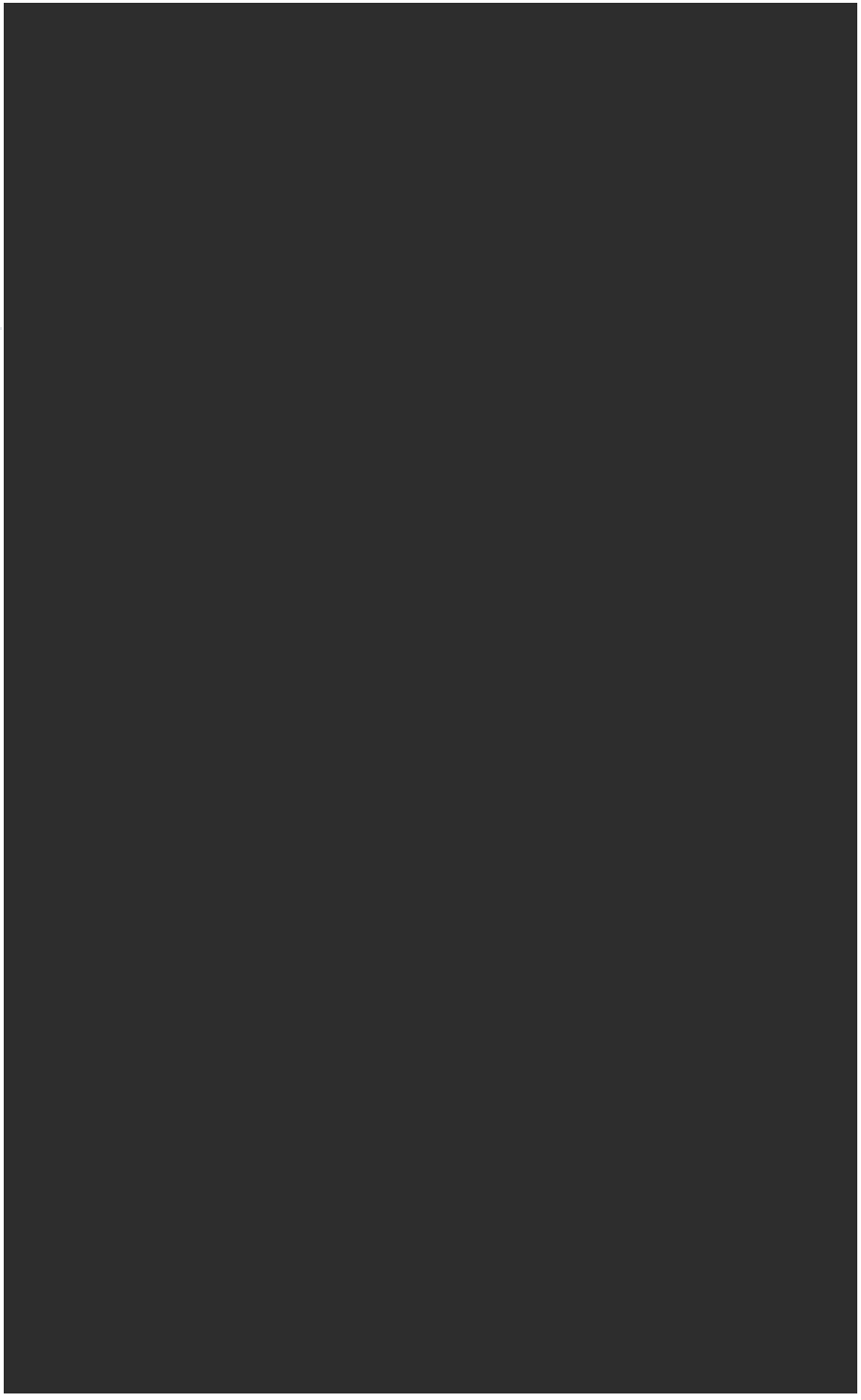
To make API requests, you need authenticate to Upwork API. Currently, we support OAuth 1.0 authentication. All API requests MUST be signed following the **RFC 5849** specification.

OAuth 1.0

The OAuth protocol enables websites or applications (clients) to access protected resources from a web service (server) via an API, without requiring resource owners to disclose their service provider credentials to the clients. For more information on the OAuth workflow process visit Beginner's Guide to OAuth and the OAuth 1.0 Guide.

Client credentials

For each application you develop, you need to obtain new client credentials. These include a client identifier and a

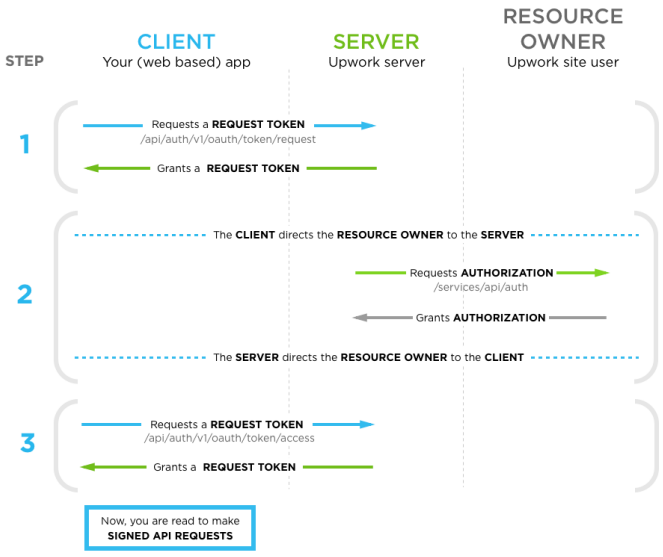


client shared-secret. You can request these credentials at https://www.upwork.com/services/api/apply while logged into your Upwork account. You will receive a public and a private key for each client identifier and client shared-secret you request.

Upwork OAuth 1.0 workflow

At a basic level, the OAuth 1.0 authentication process involves the following steps:

1. Get Request token
2. Get authorization from the **resource owner** and obtain a **Verifier** code
3. Exchange **Request token** and **Verifier** for an **Access Token**



Below you will find the list of API calls needed for

authentication.

Get request token

Endpoint

POST /api/auth/v1/oauth/token/request

HTTPS is required. All the names of variables follow OAuth specification (see RFC 5849).

Required key permissions

No special permissions are required

Parameters

oauth_signature:	**required, string** OAuth signature.
oauth_consumer_key:	**required, string** A consumer (application) key.
oauth_nonce:	**required, string** OAuth nonce.
oauth_timestamp:	**required, string** OAuth timestamp.
oauth_signature_method:	**optional, string** Signature method (persistent constant equal to 'HMAC-SHA1').
oauth_callback:	**optional, string** Callback URL used after confirming Authorization.

DEFINITION

There's no public method for getting request token in node-upwork, request token is retrieved during invocation of the authorization:

```
api.getAuthorizationUrl(callbackUrl,
callback);
```

EXAMPLE REQUEST

There's no public method for getting request token in node-upwork, request token is retrieved during invocation of the authorization (see below). You can do it manually using the oAuth library if necessary.

EXAMPLE RESPONSE

See the Authorization section.

Error messages

401:	Failed to validate oauth signature and token

Returns

Returns a token/secret pair to be used for authorization and getting the access token. Example:

```
oauth_callback_confirmed=1&oauth_token=d6b9dba626bf43f187a2aed2d5f8e387&oauth_token_secret=517af7d4c335cc59
```

Authorize and get verifier

Upwork server needs to request authorization from the resource owner to grant you access to the required data. To do so, you need to redirect the resource owner to Upwork server's authentication endpoint.

Endpoint

GET /services/api/auth

If the user is not logged in to Upwork, she/he is asked to do so via 'www.upwork.com/login?redir=https://www.upwork.com/services/api/auth'. Upwork checks if the user has already authorized this application. If the application is already authorized, user is redirected to the application using the callback URL provided with the

DEFINITION

```
api.getAuthorizationUrl(callbackUrl,
callback);
```

EXAMPLE REQUEST

```
var UpworkApi = require('upwork-api')
  , rl = require('readline');

var config = {
  'consumerKey' : 'FILL_ME',
  'consumerSecret' : 'FILL_ME'
};

var api = new UpworkApi(config);
var callbackUrl =
'http://my.callback.example.com';

api.getAuthorizationUrl(callbackUrl,
```

request or (if empty) the one registered for the used OAuth key. If the user has not authorized the application Upwork asks her/him to do so.

Required key permissions

Standard authorization at Upwork is required

Parameters

oauth_token:	**required, string**	This is a request token received in the first step of oAuth workflow.

Error messages

403:	oAuth token not found

Returns

If the authorization is successful, this call returns a verifier. You can also check for the `X-Upwork-Oauth-Verifier` header in response.

If the resource owner is not currently logged in to Upwork, he/she is asked to do so at `www.upwork.com/login?redir=https://www.upwork.com/services/api/auth...`.

Then, Upwork checks if the resource owner has already authorized your (web server) application.

• If the resource owner has not authorized your (web server) application, Upwork asks the resource owner if he/she wishes to do so.

```
  function(error, url, requestToken,
requestTokenSecret) {
    if (error) throw new Error('can not get
authorization url, error: ' + error);

    // Authorize application
    var i =
rl.createInterface(process.stdin,
process.stdout);
    i.question('Please, visit an url ' +
url + ' and enter a verifier: ',
function(verifier) {
      i.close();
      process.stdin.destroy();
    });
});
```

EXAMPLE RESPONSE

Get the verifier manually or extract it from the callback request.

• If your application is authorized, the resource owner is redirected to your (web server) application using the callback URL provided with the request, or, if empty, to the one registered for the client identifier used. The callback URL will have the following query parameters appended: `?oauth_token=...&oauth_verifier=...`. For desktop applications, there's no callback triggered. Instead, the verifier is shown to the resource owner in the browser window.

Get access token

Once you receive the request token and the resource owner's authorization (verifier code), you are ready to request Upwork Server an Access token.

Endpoint

POST /api/auth/v1/oauth/token/access

HTTPS is required. All the names of variables follow OAuth specification (see RFC 5849).

Required key permissions

No special permissions are required

Parameters

oauth_token: **required, string**

The temporary credentials identifier,

DEFINITION

```
api.getAccessToken(
  requestToken, requestTokenSecret,
  verifier, callback);
```

EXAMPLE REQUEST

```
var UpworkApi = require('upwork-api')
  , rl = require('readline');

var config = {
  'consumerKey' : 'FILL_ME',
  'consumerSecret' : 'FILL_ME'
};

var api = new UpworkApi(config);
var callbackUrl =
'http://my.callback.example.com';

api.getAuthorizationUrl(callbackUrl,
function(error, url, requestToken,
```

	this is the Request token received in the first step.
oauth_consumer_key:	**required, string** A consumer (application) key.
oauth_signature:	**required, string** OAuth signature.
oauth_nonce:	**required, string** OAuth nonce.
oauth_timestamp:	**required, string** OAuth timestamp.
oauth_verifier:	**required, string** OAuth verifier received in the authorization call in the previous step of the oAuth workflow.
oauth_signature_method:	**required, string** Signature method (persistent constant equal to 'HMAC-SHA1').

Error messages

401:	The consumer_key and token combination does not exist or is not enabled
401:	OAuth Verification Failed
403:	Forbidden: oauth_token not found
403:	Forbidden: oauth_signature not found
403:	Forbidden: oauth_consumer_key not found

Returns

```
requestTokenSecret) {
  if (error) throw new Error('can not get
authorization url, error: ' + error);

  // Authorize application
  var i =
rl.createInterface(process.stdin,
process.stdout);
  i.question('Please, visit an url ' +
url + ' and enter a verifier: ',
function(verifier) {
    i.close();
    process.stdin.destroy();

    // Get access token/secret pair
    api.getAccessToken(requestToken,
requestTokenSecret, verifier,
function(error, accessToken,
accessTokenSecret) {
      if (error) throw new Error(error);

      // Here you can store access token
in safe place

    });
  });
});
```

EXAMPLE RESPONSE

You get error message, access token and access token secret as arguments in your callback function.

Returns the Access token.

> **Note**
>
> Once created, the Access token never expires.

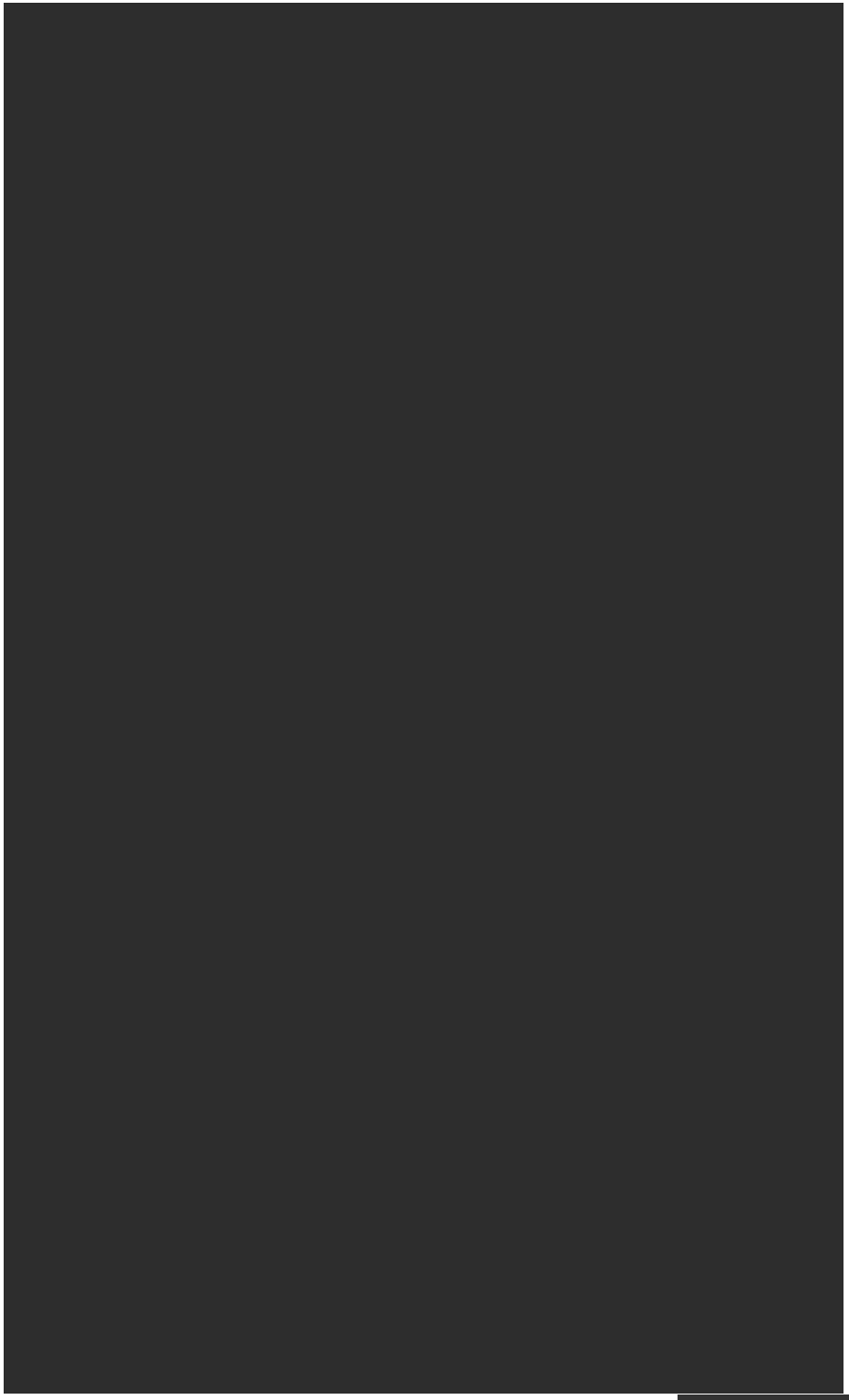
Required OAuth 1.0 parameters

Each API request must include a list of OAuth parameters (all the descriptions and meanings of parameters are defined by the OAuth specification at **RFC 5849**):

- oauth_consumer_key
- oauth_signature
- oauth_nonce
- oauth_signature_method (persistent constant equal to HMAC-SHA1)
- oauth_timestamp
- oauth_token
- any additional parameters for the specific API request.

Here is an example of a simple API request based on an authorized OAuth session:

```
/api/team/v1/teamrooms/upwork.json?
oauth_signature=jL08juSnj9FQDzHY6%2BB4yr25QiA%3D&
oauth_consumer_key=5bba83419248517d7883285e2b5976
b1&oauth_nonce=580f9c6d6b490fb16bb5dd29b5c&
oauth_signature_method=HMAC-
```

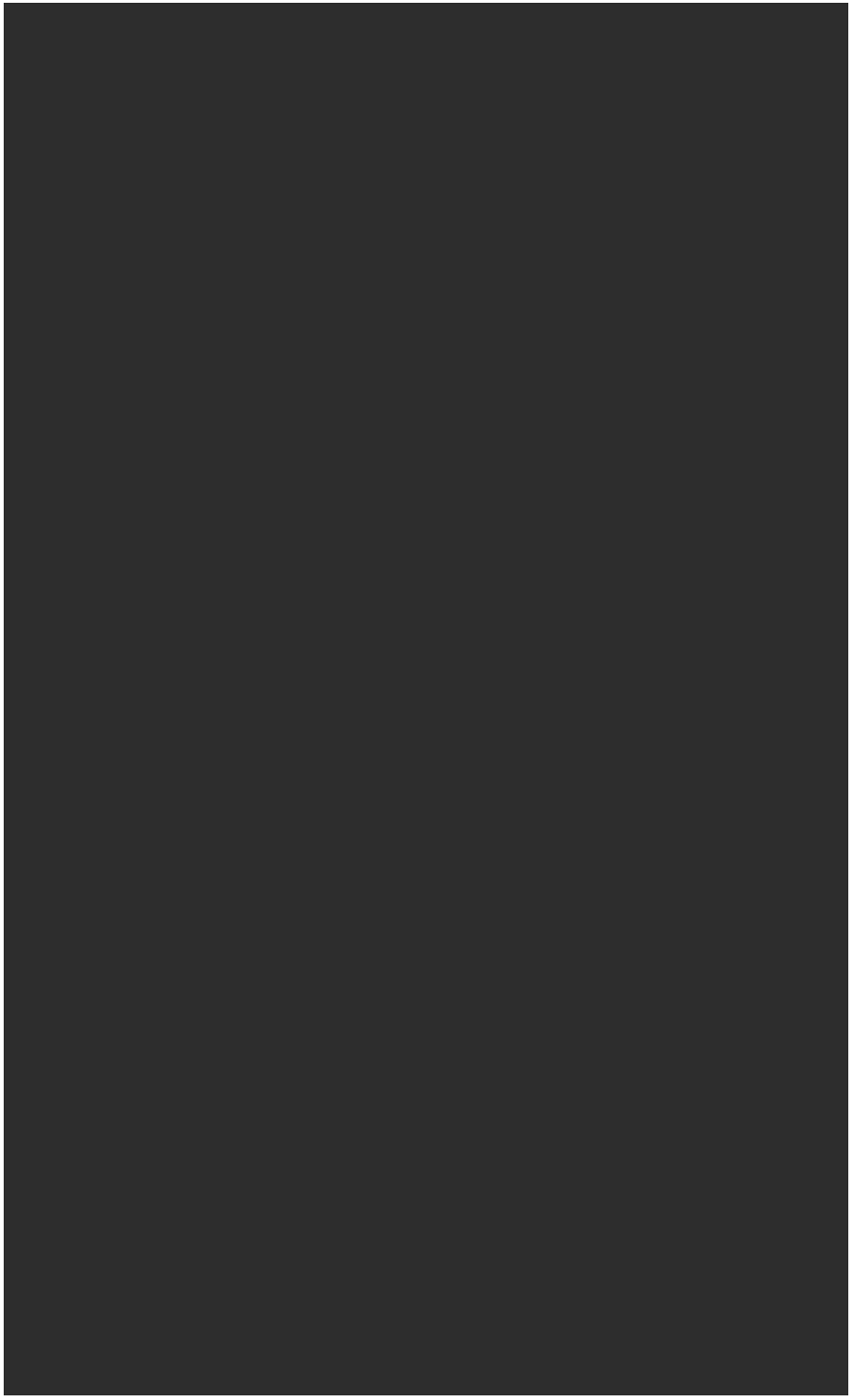


`SHA1&oauth_timestamp=1292557402&oauth_token=6b3b412b60c4e7000329e990a4dbbb2c&online=all`

Troubleshooting authentication issues

Authentication errors are the most common issues received by our Support team. Here are some troubleshooting tips you may want to try to solve frequent authentication issues and start using our API.

• Most OAuth signing issues are caused by an invalid format in the signature base string.
• If you use a header-based OAuth request, make sure that the HTTP authorization header is being properly setup and formatted according to the specific language you use. Remember that you SHOULD NOT repeat any of the `oauth_*` parameters in the POST body or URL of the request you are making. Parameters that begin with `oauth_*` SHOULD NOT be part of the POST body or query string.
• Don't include unnecessary `oauth_*` parameters in the request.
• Use `auth` in all REST API methods that support it. All Upwork REST API methods require authentication and using `auth` ensures that the requests are evaluated within the context of your current user.
• Use valid endpoints.



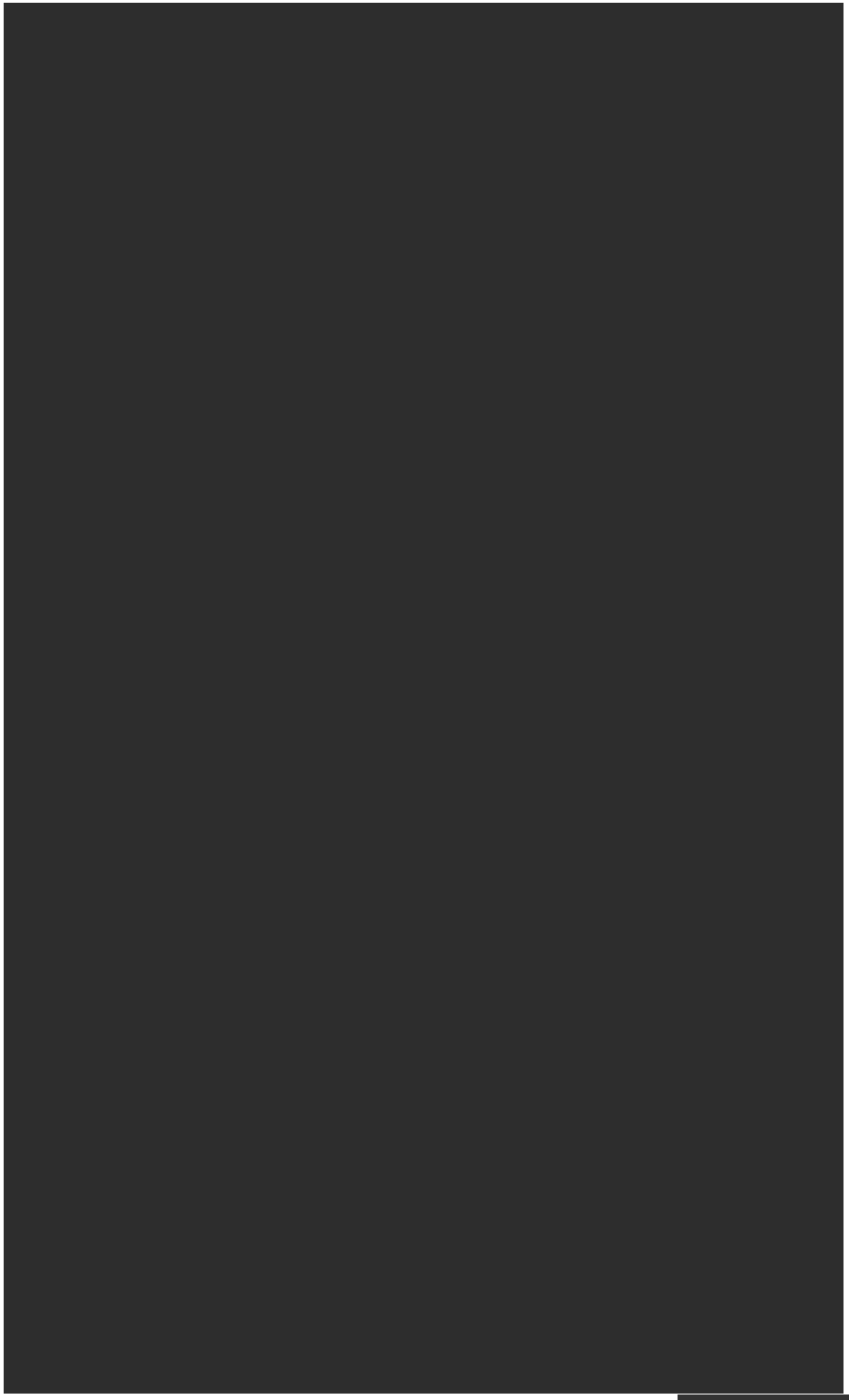
• Make sure you are using the appropriate HTTP request method. Most calls to the Upwork API use `POST` or `GET` methods.
• If you use an OAuth library, make sure you select a well-supported one.
• Try making the request in another OAuth library or tool. Comparing successful and failed requests will help you identify what might be going wrong.
• Learn how to override the `oauth_timestamp` and `oauth_nonce` values in your OAuth library. Use this capability to replay signature generation scenarios for comparative analysis.
• `oauth_token` and `oauth_token_secret` strings change when a user's access moves between permission levels, or if a user denies your application access and then re-grants access. Never assume that the strings will remain constant.

The following are the most frequent authentication errors you may come across:

400 - Bad request

The request could not be understood by the server due to malformed syntax.

Suggestion: Review the error message and make the necessary modifications before retrying. You should not repeat the request without modifications.



• Can’t verify request, missing `oauth_consumer_key` or `oauth_token`.
• Can’t verify request signature, missing parameter.
• None of the signing methods is supported.
• Unsupported signature method.
• Expected OAuth version.
• Unsupported characters in host name.
• Unsupported scheme type, expected Timestamp is out of sequence.

401 - Unauthorized

User authentication is required and has failed or has not yet been provided.

Suggestion: Check the procedure for signature calculation.

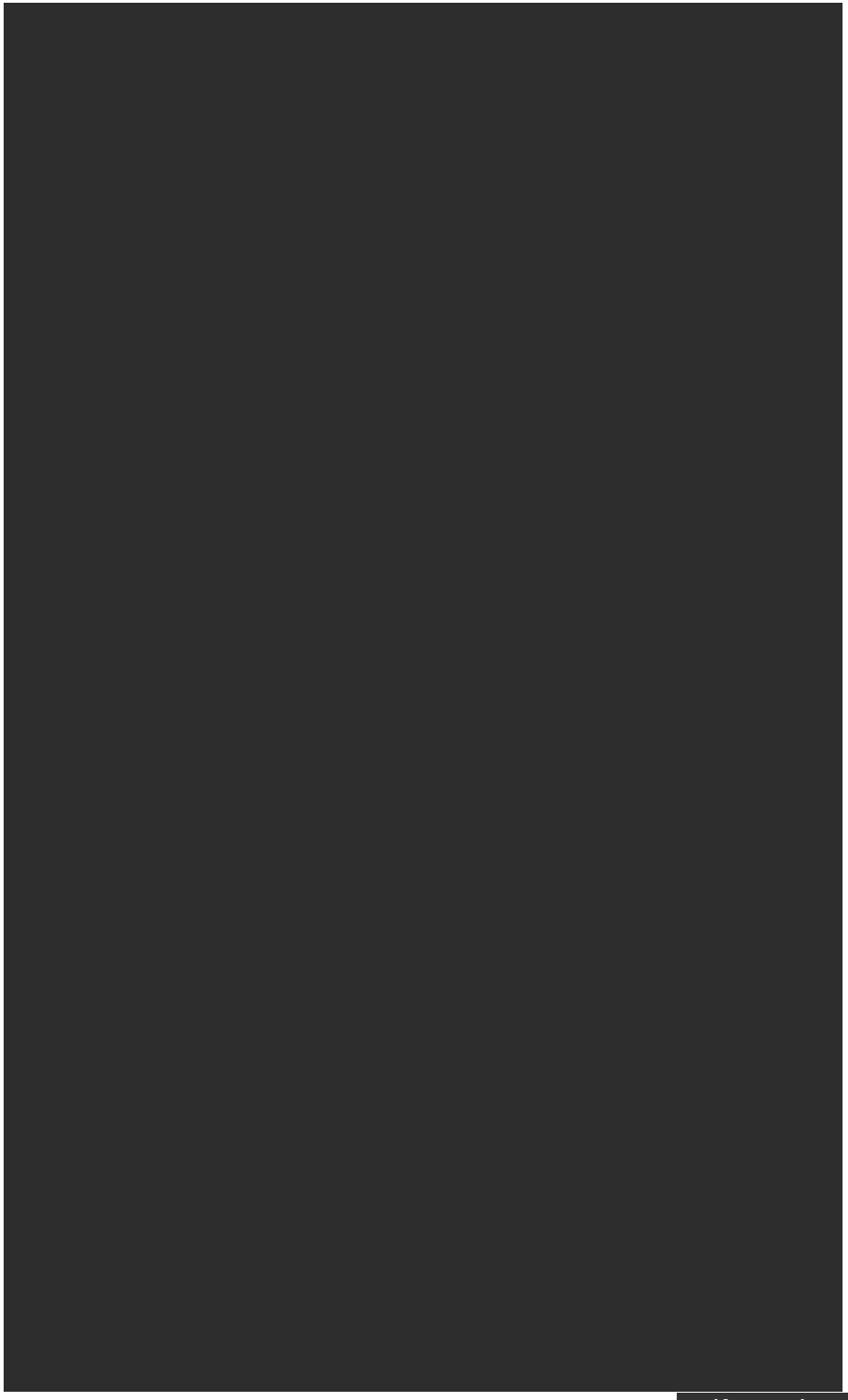
• Verification of signature failed.

403 - Forbidden

The server understood the request, but is refusing to fulfill it.

Suggestion: Review the error message and make the necessary modifications before retrying. You should not repeat the request without modifications.

• Duplicate timestamp/nonce combination, possible replay attack.



• The consumer_key and token combination does not exist or is not enabled.

Users

This section describes resources that return information about users.

Authenticated User

Endpoint

GET /api/auth/v1/info.{format}

This API call returns detailed information about the currently authenticated user.

Required key permissions

Access your basic info

Arguments

format:	**optional, string**
	Default: `xml`
	Valid values: `json, xml`
	Response format.

Error messages

403:	Forbidden

DEFINITION

```
auth.getUserInfo(callback);
```

EXAMPLE REQUEST

```
var Auth = require('upwork-
api/lib/routers/auth').Auth;

var auth = new Auth(api);
auth.getUserInfo(function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'server_time': '1400662109',
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Bangkok',
```

Returns

Returns information about the currently authenticated user.

```
      'timezone_offset': '25200'
    },
    'info': {
      'portrait_50_img': 'https://...',
      'ref': '525',
      'portrait_32_img': 'https://...',
      'has_agency': '0',
      'portrait_100_img': 'https://...',
      'company_url': '',
      'capacity': {
        'provider': 'yes',
        'buyer': 'yes',
        'affiliate_manager': 'no'
      },
      'location': {
        'city': 'San Francisco',
        'state': 'CA',
        'country': 'United States'
      },
      'profile_url': 'https://...'
    }
  }
```

Referenced User

Endpoint

GET /api/hr/v2/users/{user_reference}.{format}

This API call returns details about the referenced user.

Required key permissions

View the structure of your companies/teams

Arguments

user_reference: **required, string**

The reference ID of the user or

DEFINITION

```
users.getMyInfo(callback);
users.getSpecific(userReference,
callback);
```

EXAMPLE REQUEST

```
var Users = require('upwork-
api/lib/routers/organization/users.js').U
sers;

var users = new Users(api);
users.getMyInfo(function(error, data) {
```

The reference ID of the user or special value `me` to get information about the currently authenticated user. Examples: `me`, `33333`.

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Error messages

401: Unauthorized

403: Forbidden

Returns

Returns the following information about the referenced user:

reference: **reference to User object**

The reference ID of the user

id: **string**

The literal ID of the user

first_name: **string**

The user's first name

last_name: **string**

The user's last name

timezone: **string**

The user's time zone.

timezone_offset: **integer**

The time zone offset in seconds

```
    console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'timezone': 'UTC+07:00 Bangkok,
Jakarta, Hanoi',
  'status': 'active',
  'timezone_offset': '25200',
  'public_url':
'https://www.upwork.com/...',
  'last_name': 'Johnson',
  'email':
'my_upwork_username@example.com',
  'reference': '12345',
  'id': 'my_upwork_username',
  'is_provider': '1',
  'first_name': 'John',
  'profile_key': '...'
}
```

is_provider: **boolean**

Indicates whether the user is a freelancer (can be hired) or not

status: **string (“active”, “inactive”)**

The status of the user

User Permissions

Endpoint

GET /api/hr/v2/userroles.{format}

This API call returns detailed information about the currently authenticated user including some data from his/her profile.

Required key permissions

View the structure of your companies/teams

Arguments

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Error messages

401: Unauthorized

403: Forbidden

Returns

While permission and role information is contained in the `permision` field, this API returns a complete list of a team

DEFINITION

```
roles.getAll(callback);
roles.getBySpecificUser(reference,
callback);
```

EXAMPLE REQUEST

```
var Roles = require('upwork-
api/lib/routers/hr/roles.js').Roles;

var roles = new Roles(api);
roles.getAll(function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'userrole': [
    {
      'parent_team__id':
'0123456789abcdf',
      'user__first_name': 'John',
      'permissions': {
        'permission': [
          'manage_finance',
          'manage_employment',
```

information for each team that the authorized user has access to. It is also important to consider that a user may have multiple roles within a team.

For example, a user who has full manager permissions within a team returns the following fields:

manage_finance:	Access to the company or team financial details.
manage_affiliation:	Access to the affiliate freelancers within a company or team.
manage_employment:	Permission to start engagements (actually hire freelancers).
manage_recruiting:	Permission to post new jobs and interview candidates.

```
            'manage_recruiting',
            'manage_teamroom'
          ]
        },
        'company__reference': '12345',
        'user__last_name': 'Johnson',
        'team__is_hidden': '',
        'reference': '12345',
        'team__reference': '12345',
        'affiliation_status': 'none',
        'user__reference': '12345',
        'user__is_provider': '1',
        'parent_team__name': 'Upwork',
        'has_team_room_access': '1',
        'parent_team__reference': '12345',
        'team__id': '0123456789abcdf',
        'engagement__reference': '',
        'team__name': 'Upwork',
        'company__name': 'Upwork',
        'role': 'admin',
        'user__id': 'my_user_id',
        'is_owner': '1'
      },
      # ...
    ]
  }
```

DEFINITION

Public profiles

Public profiles resources allow you to search freelancer's public profiles and get detailed information on a specific profile.

Search for Freelancers

Endpoint

GET /api/profiles/v2/search/providers.{format}

The search parameters include the options available on the site and additional options to configure the format of your results.

Required key permissions

No special permissions are required

Arguments

format: **optional, string**

Default: `json`

Valid values: `json`

Response format.

Parameters

q: **conditionally required, string**

The search query. At least one of the `q`, `title`, `skill` parameters should be specified.

title: **conditionally required, string**

Searches for the title in the freelancer's profile. At least one of the `q`, `title`, `skill` parameters should be specified.

skills: **conditionally required, string**

Searches for skills of freelancer's profile. At least one of the `q`, `title`, `skill` parameters should be

```
freelancers.find(params, callback);
```

EXAMPLE REQUEST

```
var Search = require('upwork-
api/lib/routers/freelancers/search.js').S
earch;

var freelancers = new Search(api);
var params = {'q': 'python', 'title':
'Web Developer'};
freelancers.find(params, function(error,
data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
[
 {'categories2': ['Legal',
                  'Web & Mobile
Development',
                  'Admin Support'],
  'country': 'India',
  'description': 'I do ...',
  'feedback': '4.8424790960452',
  'id': '~aaaa9999d3f394624e',
  'last_activity': 'June 17, 2014',
  'member_since': 'July 21, 2011',
  'name': 'John Johnson',
  'portfolio_items_count': '1',
  'portrait_50': 'https://...',
  'profile_type': 'Independent',
  'rate': '22.22',
  'skills': ['python',
             'django-framework',
             'mongodb',
             'jquery',
             'html5',
             'postgresql'],
```

specified.

groups: **optional, string**

Searches for groups in the freelancer's profile. The freelancer must be a member of the group provided.

tests: **optional, string**

Searches for tests in the freelancer's profile.

tests_top_10: **optional, string**

Searches for freelancers that are in the top 10 of the test.

tests_top_30: **optional, string**

Searches for freelancers that are in the top 30 of the test.

category2: **optional, string**

The category (V2) of the freelancer's profile. Use Metadata resource to get it. You can get it via Metadata Category (v2) resource.

subcategory2: **optional, string**

The subcategory of the job according to the list of Categories 2.0. Example: `Web & Mobile Development`. You can get it via Metadata Category (v2) resource.

region: **optional, string**

Searches for profiles of freelancers

```
    'test_passed_count': '3',
    'title': 'Web Developer'},
    {
      # Another freelancer
    },
    # ...
  ]
```

who live in the region provided. Valid values are provided by Metadata Regions resource. Example: `Latin America`.

feedback: **optional, string**

Searches for freelancers with specific feedback score. Single values such as `3` or `3,4` (comma-separated values results in OR queries) and ranges such as `[3 TO 4]` are valid.

rate: **optional, string**

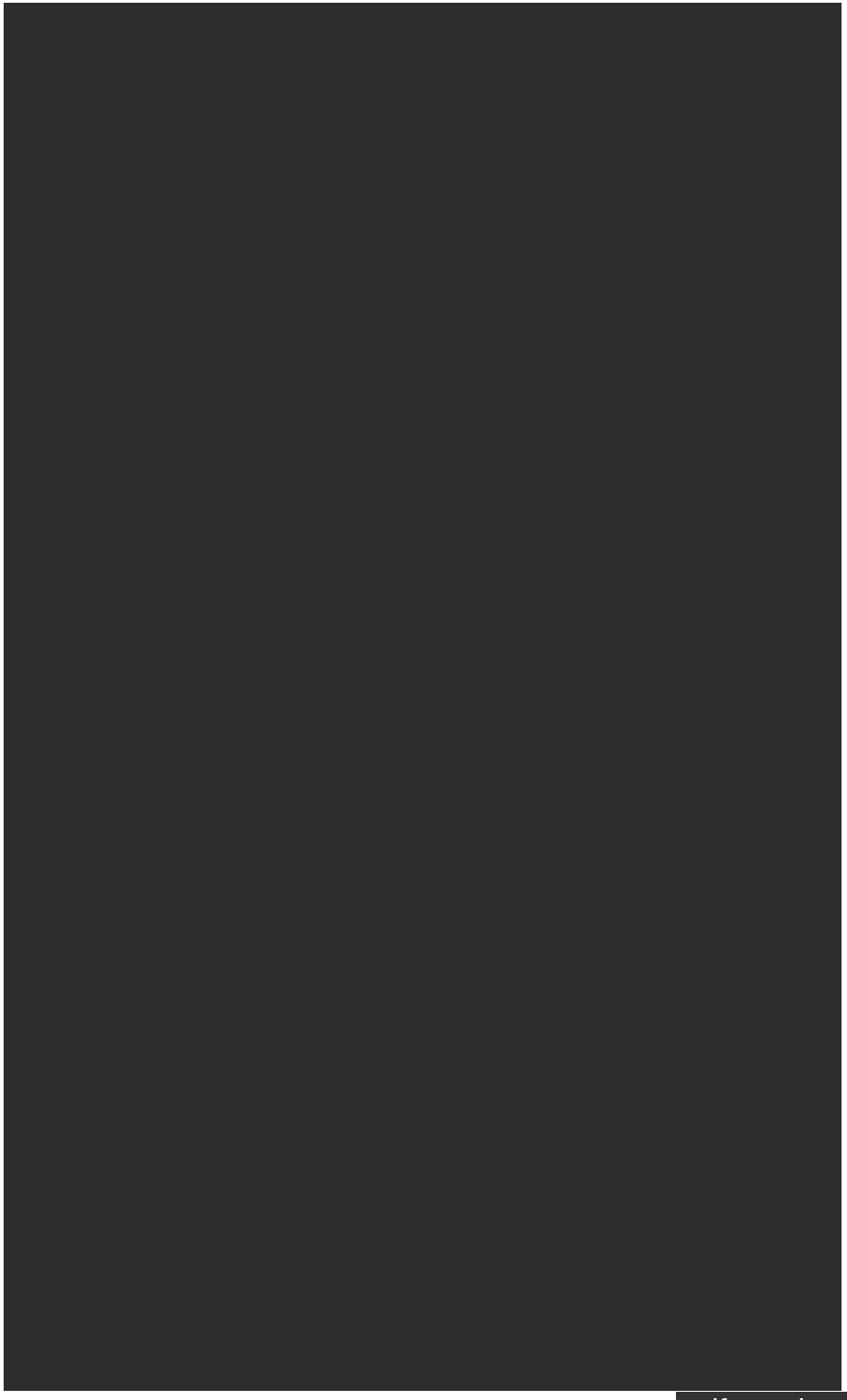
A number or range used to filter the search by freelancer's profile rate. Single values such as `20` or `20,30` (comma-separated values result in `OR` queries) and ranges such as `[20 TO 40]` are valid.

hours: **optional, string**

Searches for profiles of freelancers who have worked the number of hours provided. Single values such as `20` or `20,30` (comma-separated values result in `OR` queries) and ranges such as `[20 TO 40]` are valid.

recent_hours: **optional, string**

Searches for profiles of freelancers who have recently worked the number of hours provided. Single values such



as `20` or `20,30` (comma-separated values result to `OR` queries) and ranges such as `[20 TO 40]` are valid.

last_activity: **optional, string**

The date of the last time the freelancer worked. The value should be formatted according to ISO 8601 date syntax with hours always set at 00:00:00.000. Example: `2013-01-04T00:00:00.000Z`.

english_skill: **optional, integer**

Valid values: `0, 1, 2, 3, 4, 5`

The freelancer's English skills assessment result.

is_odesk_ready: **optional, integer**

Valid values: `0, 1`

Defines if the freelancer is Upwork ready or not.

profile_type: **optional, string**

Valid values: `Agency, Independent`

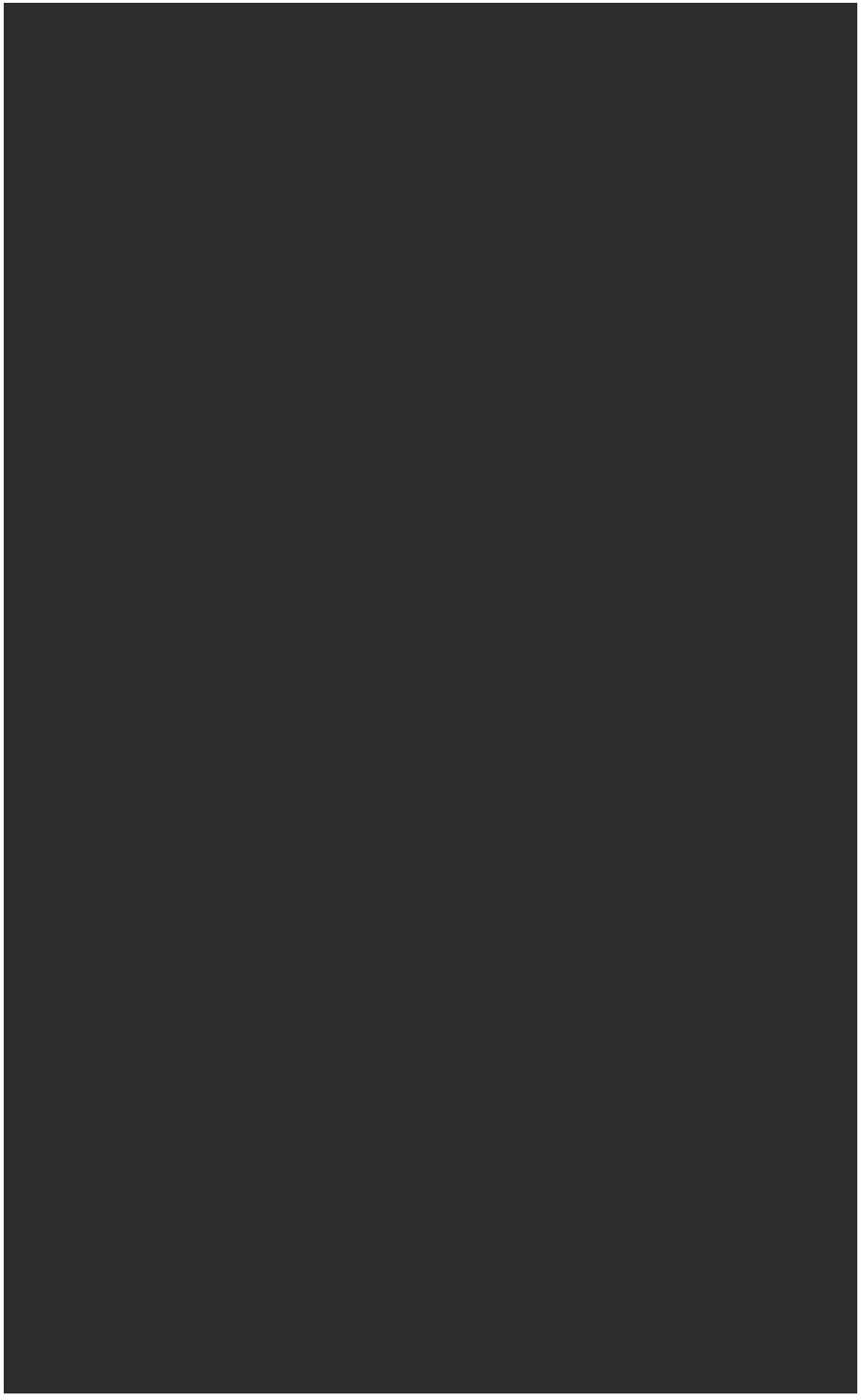
The freelancer type.

include_entities: **optional, string**

Valid values: `0, 1`

If set to `1` the data in the response will only contain a profile IDs array.

profile_access: **optional, string**



Valid values: `auth, public`

The access type of the profile.

nss100: **optional, string**

Job success score. Example: value `[0.8 TO *]` returns freelancers with job success score more then 0.8 (80%).

paging: **optional, string**

Default: `0;10`

Pagination, formed as `$offset;$count`. Page size is restricted to be <= 100. Example: page=100;99.

Error messages

401: Unauthorized

Possible output fields

(Click here to show/hide)

Returns

Returns the list of the objects with information about each freelancer who matches the requested query and parameters.

Get brief profile summary

Endpoint

GET /api/profiles/v1/providers/{profile_key}/brief.{format}

DEFINITION

```
freelancers.getSpecificBrief(key,
callback);
```

Required key permissions

No special permissions are required

Arguments

profile_key: **required, string**

The freelancer's profile key or a list of keys, separated by semicolon (';'). The number of keys per request is limited to 20.

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Error messages

401: Unauthorized

413: Request Entity Too Large

Returns

Returns a brief summary of a freelancer profile.

EXAMPLE REQUEST

```
var Profile = require('upwork-
api/lib/routers/freelancers/profile.js').
Profile;

var freelancers = new Profile(api);
freelancers.getSpecificBrief('~aa9955d3f3
94624e', function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{'ciphertext': '~aaa99955d3f394624e',
 'dev_ac_agencies': '',
 'dev_adj_score': '4.8424790960452',
 'dev_adj_score_recent':
'4.83652565536548',
 'dev_billed_assignments': '16',
 'dev_city': 'Bangkok',
 'dev_country': 'Thailand',
 'dev_eng_skill': '5',
 'dev_groups': '',
 'dev_is_affiliated': '0',
 'dev_last_activity': 'June 17, 2014',
 'dev_last_worked': 'June 8, 2014',
 'dev_last_worked_ts': '1402185600000',
 'dev_portfolio_items_count': '1',
 'dev_portrait': 'https://...',
 'dev_portrait_100': 'https://...',
 'dev_portrait_32': 'https://...',
 'dev_portrait_50': 'https://...',
 'dev_profile_title': 'Web Developer',
 'dev_recno_ciphertext':
'~aaa99955d3f394624e',
 'dev_short_name': 'John Johnson',
 'dev_timezone': 'UTC+07:00 Bangkok,
Jakarta, Hanoi',
 'dev_tot_feedback': '10',
```

Get freelancer profile by key

Endpoint

GET /api/profiles/v1/providers/{profile_key}.{format}

The API takes a profile key and returns detailed profile information about a freelancer or list of freelancers. This call returns an exhaustive list of attributes associated with the freelancer. If you're looking for a shorter summary, use the brief version of this call (GET /api/profiles/v1/providers/{profile_key}/brief.{format}).

Required key permissions

No special permissions are required

Arguments

profile_key:	**required, string** The freelancer's profile key or a list of keys, separated by semicolon (';'). The number of keys per request is limited to 20.
format:	**optional, string** **Default:** `xml` **Valid values:** `json, xml` Response format.

```
  'dev_total_hours': '48.6666666666667',
  'dev_ui_profile_access': 'Public'}
```

DEFINITION

```
freelancers.getSpecific(key, callback);
```

EXAMPLE REQUEST

```
var Profile = require('upwork-
api/lib/routers/freelancers/profile.js').
Profile;

var freelancers = new Profile(api);
freelancers.getSpecific('~aa9955d3f394624
e', function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
    'assignments': {
      # ...
    },
    'ciphertext': '~~aaa9999db7b5808b',
    'dev_ac_agencies': '',
    'dev_adj_score': '4.99924101793537',
    'dev_adj_score_recent':
'4.99981986177941',
    'dev_billed_assignments': '27',
    'dev_blurb': "My description",
    'dev_city': 'Bangkok',
    'dev_country': 'Thailand',
    'dev_eng_skill': '5',
```

Error messages

401:	Unauthorized
413:	Request Entity Too Large

Possible output fields

(Click here to show/hide)

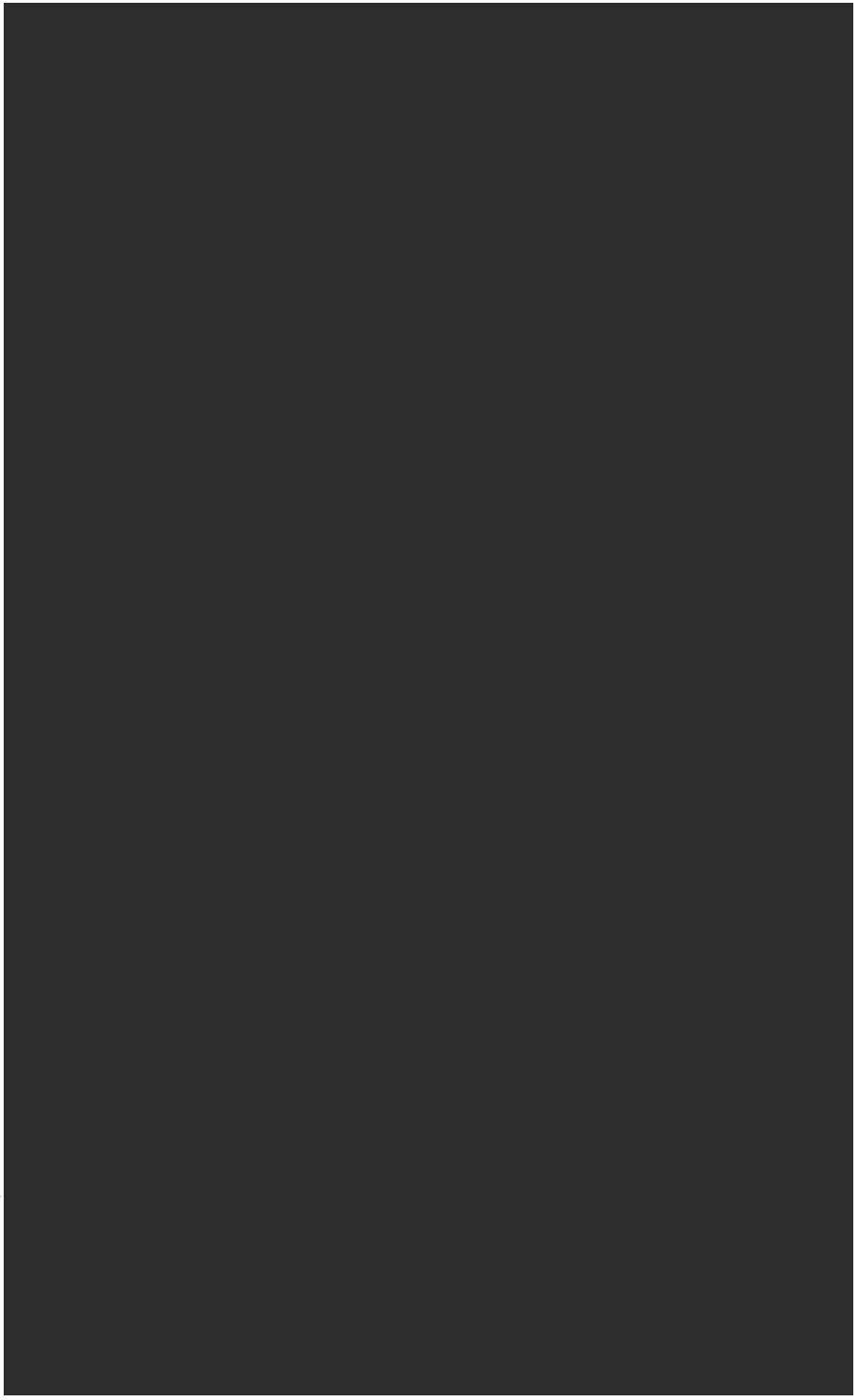
Returns

There is a lot of information returned by this call and the best way to see what it actually means is pull data from a few providers and match up the fields. Most of the response fields are human readable. Here is some additional info that should help understand this response:

dev_profile_access:	The status of the freelancer's profile (public or private).
skill:	Describes a skill that the freelancer has listed in his/her profile.
tsexam:	Describes a test that the freelancer has taken and made public.
dev_score:	Describes feedback that the freelancer has received after a job has been completed (or just closed).
dev_recent_rank_percentile:	The freelancer's rank at Upwork based on data from the last 90

```
    'dev_groups': {
      # ...
    },
    'dev_is_affiliated': '0',
    'dev_last_activity': 'June 20, 2014',
    'dev_last_worked': 'June 20, 2014',
    'dev_last_worked_ts': '1403222400000',
    'dev_portfolio_items_count': '11',
    'dev_portrait': 'https://...',
    'dev_portrait_100': 'https://...',
    'dev_portrait_32': 'https://...',
    'dev_portrait_50': 'https://...',
    'dev_profile_title': 'Python web
developer +',
    'dev_recno_ciphertext':
'~aaa999e7116ea7c92d',
    'dev_short_name': 'John Johnson',
    'dev_timezone': 'UTC+07:00 Bangkok,
Jakarta, Hanoi',
    'dev_tot_feedback': '16',
    'dev_total_hours': '48.5',
    'dev_ui_profile_access': 'Public',
    'education': {
      # ...
    },
    'experiences': {
      # ...
    },
    'job_categories': {
      # ...
    },
    'permalink': '',
    'portfolio_items': {
      # ...
    },
    'skills': {
      # ...
    },
    'tsexams': {
      # ...
    }
  }
```

	days.
dev_active_interviews:	The number of active interviews the freelancer is engaged in at the moment.
dev_total_hours:	The total hours worked on Upwork by the freelancer.
experience:	Describes the freelancer's experience as listed in his/her profile under the Experience section.
assignment:	Describes past assignments that are publicly viewable.
skill, skl_level:	A skill level for the current freelancer. Values can be one of the following: `Familiar`, `Good`, `Very Good`, `Proficient`, `Expert`.
dev_ic, affiliated:	Both these metrics return the same data. They were added at different times and both of them are kept for backward compatibility purposes.



Jobs

This section describes API resources to manage jobs and

job-related activities. You can use these resources to search for jobs, post new jobs, list posted jobs, update jobs and delete them. You can also invite freelancers to a job interview.

Search for jobs

Endpoint

GET /api/profiles/v2/search/jobs.{format}

The search parameters mirror the options available on the site plus options to configure the format of your results.

Required key permissions

No special permissions are required

Arguments

format: **optional, string**

Default: `json`

Valid values: `json`

Response format.

Parameters

q: **conditionally required, string**

The search query. At least one of the `q`, `title`, `skill` parameters should be specified.

title: **conditionally required, string**

Searches for the title of the job's

DEFINITION

```
jobs.find(params, callback);
```

EXAMPLE REQUEST:

```
var Search = require('upwork-
api/lib/routers/jobs/search.js').Search;

var jobs = new Search(api);
jobs.find(params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
[
  {'budget': 750,
   'category2': 'Web & Mobile
Development',
   'client': {'country': None,
              'feedback': 0,
              'jobs_posted': 1,
              'past_hires': 0,

'payment_verification_status': None,
              'reviews_count': 0},
   'date_created': '2014-06-
30T23:50:17+0000',
```

profile. At least one of the `q`, `title`, `skill` parameters should be specified.

skills: **conditionally required, string**

Searches for skills in the job's profile. At least one of the `q`, `title`, `skill` parameters should be specified.

category2: **optional, string**

The category (V2) of the freelancer's profile. Use Metadata resource to get it. You can get it via Metadata Category (v2) resource.

subcategory2: **optional, string**

The subcategory of the job according to the list of Categories 2.0. Example: `Web & Mobile Development`. You can get it via Metadata Category (v2) resource.

job_type: **optional, string**

Valid values: `hourly, fixed-price`

The type of the Job.

duration: **optional, string**

Valid values: `week, month, quarter, semester, ongoing`

The duration of the job.

workload: **optional, string**

```
    'duration': None,
    'id': '~aaa9992d99e35a386e',
    'job_status': 'Open',
    'job_type': 'Fixed',
    'skills': ['css',
               'css3',
               'database-design',
               'database-programming',
               'english',
               'html',
               'javascript',
               'mysql',
               'php',
               'python'],
    'snippet': u"Need a custom website
<...>",
    'subcategory2': 'Web Development',
    'title': 'Looking for highly skilled
web developer',
    'url': 'http://...',
    'workload': '30+ hrs/week'},
   {
     # Another job
     # ...
   },
   # ...
 ]
```

Valid values: `as_needed, part_time, full_time`

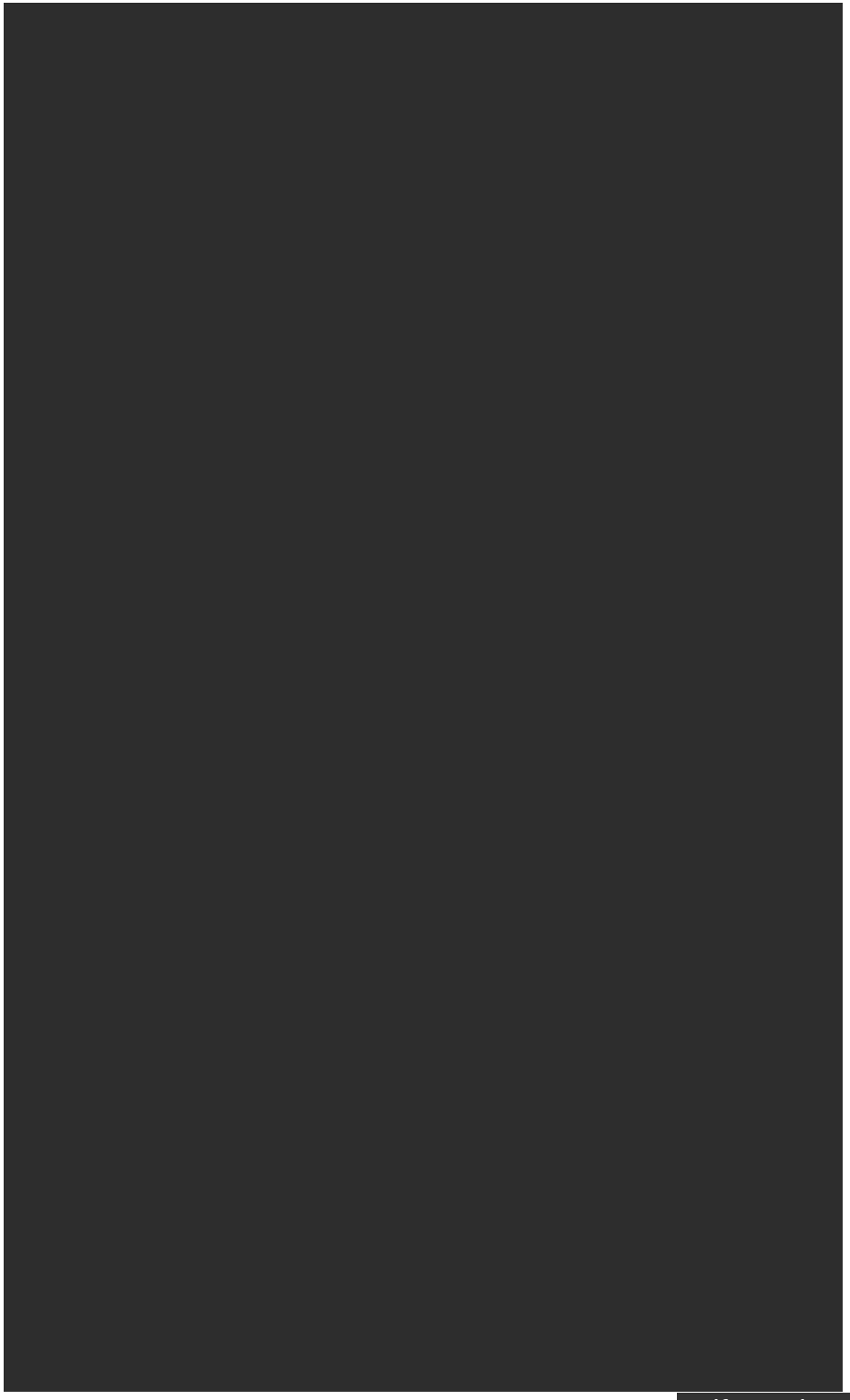
Indicates the workload for the job.

client_feedback: **optional, string**

A number or range used to filter the search by jobs posted by clients with a rating equal to, more or less than, or within the values provided. If the value is `None`, then jobs from clients without rating are returned. Single parameters such as `1` or `2,3` are valid (comma separated values result in `OR` queries). Ranges such as `[2 TO 4]` are also valid. Examples: `5.0` - the rating is equal to 5.0; `1-5` - the rating is so that 1 <= n <= 5; `1-` - the rating is >=1; `-5` - the rating is <= 5.

client_hires: **optional, string**

A number or range used to filter the search by clients with a number of past hires equal to, more or less than, or within the values provided. Single parameters such as `1` or `2,3` are valid (comma-separated values result in `OR` queries). Ranges such as `[10 TO 20]` are also valid. Examples: `5` - the number of



past hires is to 5; `0-10`: number of past hires is 0 <= n <= 10; `10-` - the number of past hires is >=10; `-5` - the number of past hires is <= 5.

budget: **optional, string**

A number or range used to filter the search by jobs having a budget equal to, more or less than, or within the values provided. For example: `[100 TO 1000]` - the budget is between 100 and 1000; `1000` - the budget is equal to 1000. `500-1000` - the budget `b` is 500 <= b <= 1000, `1000-` - the budget is >=1000; `-200` - the budget is <= 200.

job_status: **optional, string**

Valid values: `open, completed, cancelled`

The current status of the Job.

days_posted: **optional, integer**

Number of days since the job was posted.

paging: **optional, string**

Default: `0;10`

Pagination, formed as `$offset;$count`. Page size is restricted to be <= 100. Example: page=20;10.

sort: optional, string

Default: `create_time desc`

Valid values: `create_time desc`, `Client_rating desc`, `Client_total_charge desc`, `Client_total_hours desc`, `score desc`, `workload desc`, `duration desc`, `create_time asc`, `Client_rating asc`, `Client_total_charge asc`, `Client_total_hours asc`, `score asc`, `workload asc`, `duration asc`

Sorts the search results by the value provided. Example: `sort=create_time%20desc`.

Error messages

401: Unauthorized

400: Bad Request - missing parameter

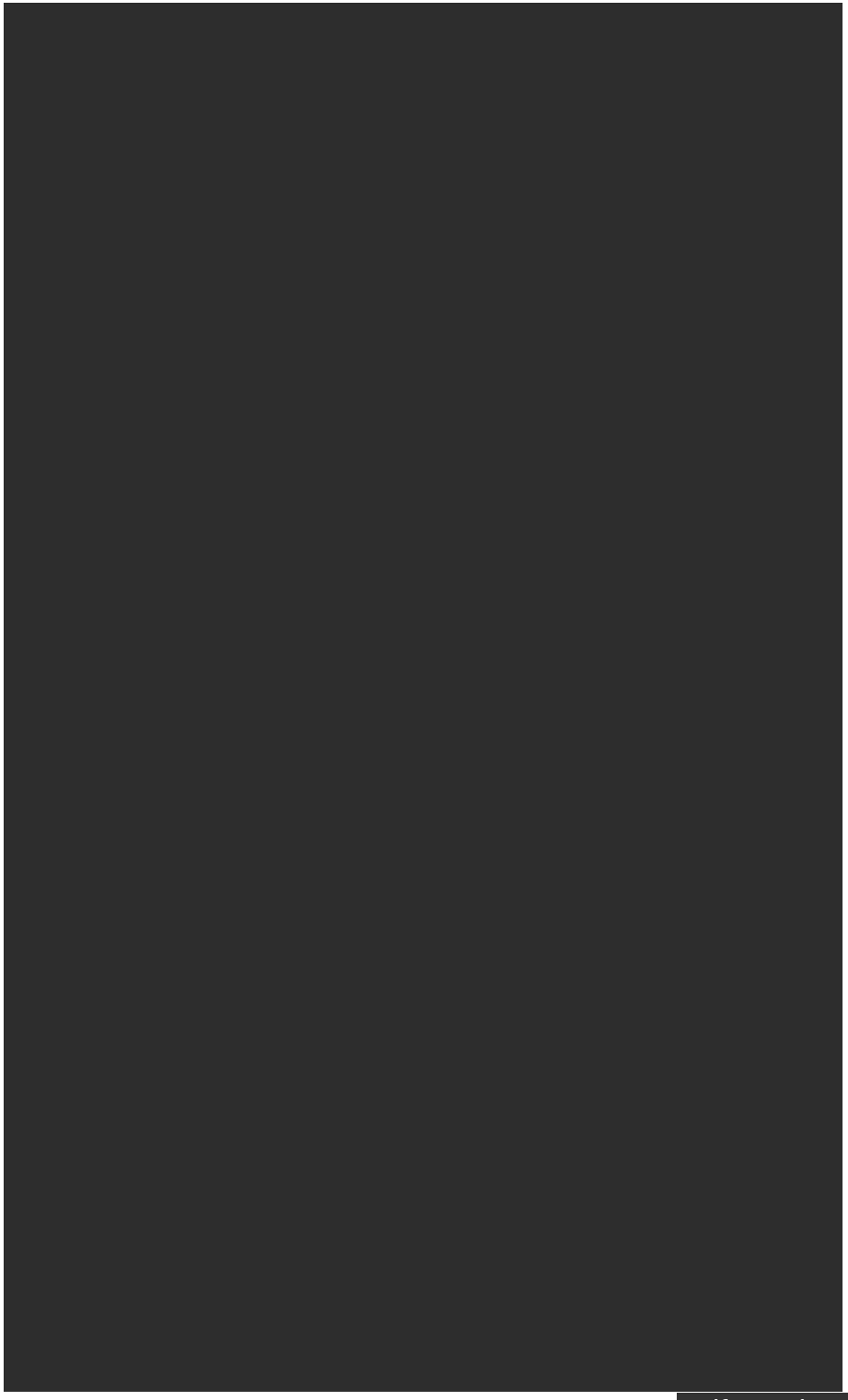
Possible output fields

(Click here to show/hide)

Returns

This resource allows third-party applications to search for public jobs on Upwork. The search parameters mirror the options available on the site.

List jobs



Endpoint

GET /api/hr/v2/jobs.{format}

This call returns all jobs that a user has `manage_recruiting` access to. This API call can be used to find the reference/key ID of a specific job.

Required key permissions

View your job posts

Arguments

format:	**optional, string** **Default:** `xml` **Valid values:** `json, xml` Response format.

Parameters

buyer_team__reference:	**required, string** The reference ID of the client's team. Example: `34567`. You can get it from List teams API call.
include_sub_teams:	**optional, string** **Valid values:** `0, 1` If set to `1`, the response includes information about sub teams.
created_by:	**optional, string** The user ID. Example: `created_by=1234`.
status:	**optional, string**

DEFINITION

```
jobs.getList(params, callback);
```

EXAMPLE REQUEST

```
jobs.getList(callback);
```

```
var Jobs = require('upwork-
api/lib/routers/hr/jobs.js').Jobs;

var jobs = new Jobs(api);
var params = {'buyer_team__reference':
'~abcdef'};
jobs.getList(params, function(error,
data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'lister': {
    'paging': {'count': '20', 'offset':
'0'},
    'query': '',
    'sort': {'sort': {'sort':
['created_time', 'asc']}},
    'total_items': '5'
  },
  'jobs': [
    {
      'attachment_file_url': '',
      'budget': '5',
      'buyer_company__name': 'My
Company',
      'buyer_company__reference':
'1040945',
      'buyer_team__name': 'My Company',
```

	Valid values: `open, filled, cancelled` The status of the job.
created_time_from:	**optional, string** Filters by 'from' time. Example: `created_time_from=2008-09-09T00:00:01`.
created_time_to:	**optional, string** Filters by 'to' time. Example: `created_time_to=2009-01-20T11:59:59`.
page:	**optional, string** Pagination, formed as `$offset;$count`. Example: `page=20;10`
order_by:	**optional, string** Sorts results by the value defined. Example: `order_by=created_time`.

Error messages

401:	Unauthorized
403:	Forbidden (operation is unauthorized)

Possible output fields

(Click here to show/hide)

Returns

If successful, the resource returns a list of jobs.

Get job by key


```
        'buyer_team__reference': '1040945',
        'cancelled_date': '1380067200000',
        'category2': 'Web, Mobile &
    Software Dev',
        'created_time': '1377423220000',
        'description': 'Testing some
    functionality',
        'duration': '',
        'end_date': '1380067200000',
        'filled_date': '',
        'job_ref_ciphertext':
    '~12345abcdf',
        'job_type': 'fixed-price',
        'keep_open_on_hire': '',
        'num_active_candidates': '0',
        'num_candidates': '0',
        'num_new_candidates': '0',
        'preference_candidate_type':
    'individuals',
        'public_url': 'https://...',
        'reference': '~12345abcdf',
        'skills': '',
        'start_date': '1377388800000',
        'status': 'cancelled',
        'subcategory2': 'Web & Mobile
    Development',
        'title': 'Test python-upwork',
        'visibility': 'invite-only'
      }
    ]
}
```

Endpoint

GET /api/hr/v2/jobs/{job_key}.{format}

This call returns the complete job object by job key. It's only available for users with `manage_recruiting` permissions within the team that the job is posted in.

Required key permissions

View your job posts

Arguments

job_key:	**required, string** Job key. For example: `~0150f0d859bb3453d0`.
format:	**optional, string** **Default:** `xml` **Valid values:** `json, xml` Response format.

Error messages

401:	Unauthorized
403:	Forbidden (operation is unauthorized)
400:	Bad request (wrongly requested data)

Possible output fields

(Click here to show/hide)

Returns

This resource returns the following details on the referenced job:

DEFINITION

```
jobs.getSpecific(String key);
```

EXAMPLE REQUEST

```
var Jobs = require('upwork-
api/lib/routers/hr/jobs.js').Jobs;

var jobs = new Jobs(api);
jobs.getSpecific('~12345abcdf',
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'attachment_file_url': '',
  'budget': '5',
  'buyer_company__name': 'My Company',
  'buyer_company__reference': '1040945',
  'buyer_team__name': 'My Company',
  'buyer_team__reference': '1040945',
  'cancelled_date': '1380067200000',
  'category2': 'Web, Mobile & Software
Dev',
  'created_time': '1377423220000',
  'description': 'Testing some
functionality',
  'duration': '',
  'end_date': '1380067200000',
  'filled_date': '',
  'job_type': 'fixed-price',
  'keep_open_on_hire': '',
  'num_active_candidates': '0',
  'num_candidates': '0',
  'num_new_candidates': '0',
  'preference_candidate_type':
```

title:	The title of the job.
job_type:	The job type.
description:	The description of the job.
public_url:	The public URL of the job.
created_time:	The time when the job was created.
created_by:	The ID of the user who created the job.
start_date:	The intended start date of the job.
end_date:	The intended end date of the job.
filled_date:	The date when the job was filled.
cancelled_date:	The date when the job was cancelled.
buyer_team__reference:	The reference ID for the client's team that posted the job.
buyer_team__id:	The name of the client's team that posted the job.
buyer_company__reference:	The reference ID for the company of the client's team.
buyer_company__name:	The name of the company of the client's team.

```
    'individuals',
      'public_url': 'https://...',
      'skills': '',
      'start_date': '1377388800000',
      'status': 'cancelled',
      'subcategory2': 'Web & Mobile
    Development',
      'title': 'Test python-upwork',
      'visibility': 'invite-only'
    }
```

visibility:	The visibility of the job.
budget:	The intended budget, if job fixed price.
duration:	The intended duration, if job is hourly tracked.
category2:	The category (V2) of the job.
subcategory2:	The subcategory (V2) of the job.
num_candidates:	The number of candidates.
num_active_candidates:	The number of active candidates.
num_new_candidates:	The number of new candidates.
status:	The status of the job.

Post job

Endpoint

POST /api/hr/v2/jobs.{format}

This call can be used to post a job.

Required key permissions

Create, modify and remove job posts

Arguments

format:	**optional, string**
	Default: `xml`
	Valid values: `json, xml`

DEFINITION

```
jobs.postJob(params, callback);
```

EXAMPLE REQUEST

```
var Jobs = require('upwork-
api/lib/routers/hr/jobs.js').Jobs;

var jobs = new Jobs(api);
var params = {
  'buyer_team__reference': '~12345abcdf',
  'title': 'Test oAuth API create job',
  'job_type': 'hourly',
  'description': 'A description',
  'visibility': 'public',
  'category2': 'Web, Mobile & Software
```

Response format.

Parameters

buyer_team__ref erence:	**required, string** The reference ID of the client's team that is posting the job. Example: `34567`. You can get it from List teams API call.
title:	**required, string** Title of the job. Example: `Development of API ecosystem`
job_type:	**required, string** **Valid values:** `hourly, fixed-price` The type of the job posted.
description:	**required, string** The job description. Example: `A new interesting start-up requires an API ecosystem`.
visibility:	**required, string** **Valid values:** `public, private, odesk, invite-only` The visibility of the job. Values description: `public` - the job is available to all users who search for jobs; `private` - the job is visible to the employer only; `odesk` - the job appears in search results only for Upwork users who are logged in;

```
Dev',
  'subcategory2': 'Web Development',
  'skills': 'python;javascript'
};
jobs.postJob(params, function(error,
data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{'auth_user': {'first_name': 'John',
               'last_name': 'Johnson',
               'timezone': 'Asia/Omsk',
               'timezone_offset':
'25200'},
'job': {'public_url': 'https://...',
'reference': '~aaa999f4c68af61ed6'},
'server_time': 1404364847}
```

`invite-only` - jobs do not appear in search and are used for jobs where the client wants to control the potential applicants.

category2: **required, string**

The category of the job according to the list of Categories 2.0. Example: `Web Development`. You can get it via Metadata Category (V2) resource.

subcategory2: **required, string**

The subcategory of the job according to the list of Categories 2.0. Example: `Web & Mobile Development`. You can get it via Metadata Category (v2) resource.

start_date: **optional, string**

The start date of the job. If the `start_date` is not included, the job defaults to starting immediately. Example: `06-15-2011`.

budget: **conditionally required, number**

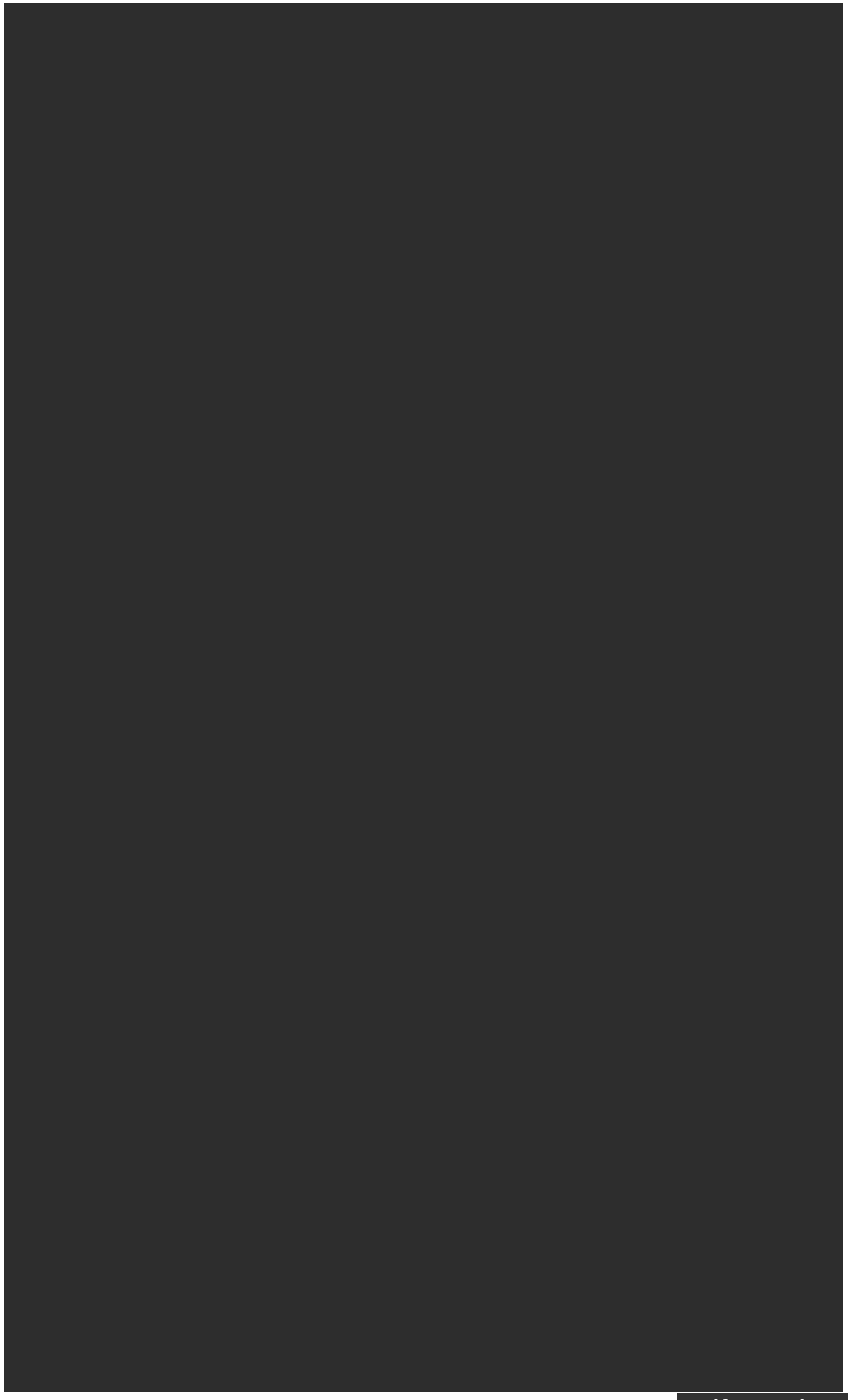
The budget for a fixed-price job. Example: `100`.

duration: **conditionally required, integer**

The duration of the job in hours. Used for hourly-jobs. Example: `90`.

skills: **optional, string**

The skills required for the job. Use



semi-colon ';' to separate the skills.

test: **optional, string**

Reference of the required test.

contractor_type: **optional, string**

Valid values: `individuals, agencies, all`

The preferred type of freelancer.

Error messages

401: Unauthorized

403: Forbidden (operation is unauthorized)

Returns

If successful, this resource returns a `200 OK` message and the job reference number.

Update job

Endpoint

PUT /api/hr/v2/jobs/{job_key}.{format}

This call updates information of a job post.

Required key permissions

Create, modify and remove job posts

Arguments

job_key: **required, string**

Job key. For example:

DEFINITION

```
jobs.editJob(key, params, callback);
```

EXAMPLE REQUEST

```
var Jobs = require('upwork-
api/lib/routers/hr/jobs.js').Jobs;

var jobs = new Jobs(api);
var key = '~abcd12345';
var params = {
  'buyer_team__reference': '~12345abcdf',
  'title': 'Test oAuth API create job',
  'description': 'Updated description',
```

`~0150f0d859bb3453d0`.

format: optional, string

Default: `xml`

Valid values: `json, xml`

Response format.

Parameters

buyer_team__reference: required, string

The reference ID of the client's team that posted the job. Example: `34567`. You can get it from List teams API call.

title: required, string

Title of the job. Example: `Development of API ecosystem`

description: required, string

The job description. Example: `A new interesting start-up requires an API ecosystem`.

visibility: required, string

Valid values: `public, private, odesk, invite-only`

The visibility of the job. Valid values are: `public` - the job is available to all users who search for jobs; `private` - the job is visible to the employer only; `odesk` - the job appears in search results only for Upwork users who are logged in;

```
    'visibility': 'public',
    'category2': 'Web, Mobile & Software
Dev',
    'subcategory2': 'Web Development',
    'duration': '100',
    'status': 'open'
};
jobs.editJob(key, params, function(error,
data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{'auth_user': {'first_name': 'John',
               'last_name': 'Johnson',
               'timezone': 'Asia/Omsk',
               'timezone_offset':
'25200'},
'job': {'message': 'updated'},
'server_time': 1404365740}
```

`invite-only` - jobs do not appear in search and are used for jobs where the client wants to control the potential applicants.

category2: **required, string**

The category of the job according to the list of Categories 2.0. Example: `Web Development`. You can get it via Metadata Category resource.

subcategory2: **required, string**

The subcategory of the job according to the list of Categories 2.0. Example: `Web & Mobile Development`. You can get it via Metadata Category (*v2*) resource. It has priority over legacy sub/category parameters.

start_date: **optional, string**

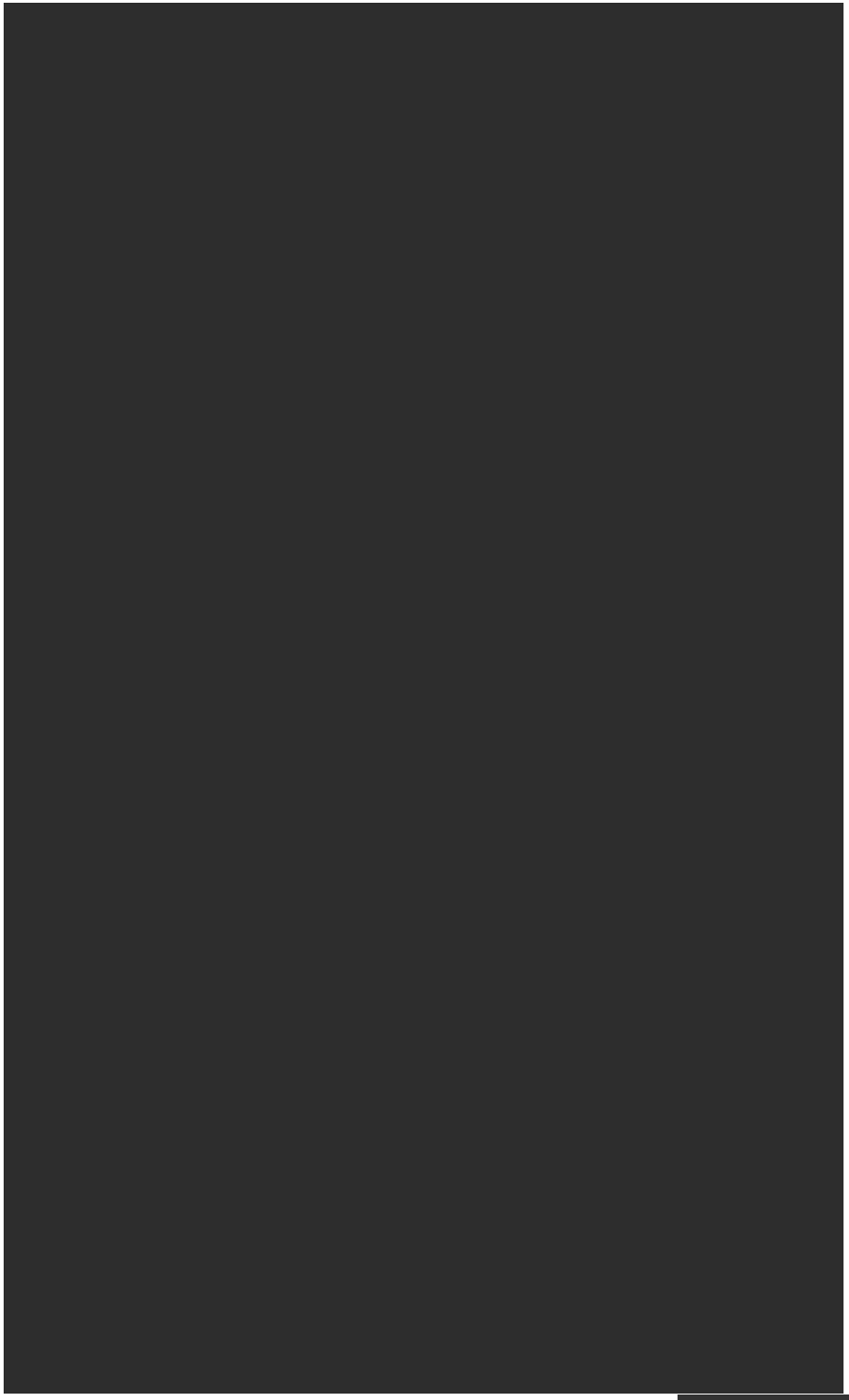
The start date of the job. If the `start_date` is not included, the job defaults to starting immediately. Example: `06-15-2011`.

budget: **conditionally required, number**

The budget for a fixed-price job. Example: `100`.

duration: **conditionally required, integer**

The duration of the job in hours. Used for hourly-jobs. Example: `90`.



status: optional, string

Valid values: `open, filled, cancelled`

The status of the job. Valid values are: `open`, `filled`, and `cancelled`.

Error messages

401: Unauthorized

403: Forbidden (operation is unauthorized)

Returns

If successful, this resource returns a 200 OK message.

Cancel job

Endpoint

DELETE /api/hr/v2/jobs/{job_key}.{format}

This call cancels a job post.

Required key permissions

Create, modify and remove job posts

Arguments

job_key: required, string

Job key. For example: `~0150f0d859bb3453d0`.

format: optional, string

Default: `xml`

DEFINITION

```
jobs.deleteJob(key, params, callback);
```

EXAMPLE REQUEST

```
var Jobs = require('upwork-
api/lib/routers/hr/jobs.js').Jobs;

var jobs = new Jobs(api);
var key = '~abcd12345';
var params = {'reason': '41'};
jobs.deleteJob(key, params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

Valid values: `json, xml`

Response format.

Parameters

reason_code: **required, string**

Valid values: `67, 51, 49, 41, 34`

The reason code to cancel the job. Valid values are: `67` - Accidental opening creation; `51` - All positions filled; `49` - Filled by alternate source; `41` - Project was cancelled; `34` - No developer for requested skills.

Error messages

401: Unauthorized

403: Forbidden (operation is unauthorized)

Returns

If successful, this resource returns a `200 OK` message.

```
{'auth_user': {'first_name': 'John',
               'last_name': 'Johnson',
               'timezone': 'Asia/Omsk',
               'timezone_offset':
'25200'},
 'job': {'message': 'deleted'},
 'server_time': 1404389886}
```

Get job profile

Endpoint

GET /api/profiles/v1/jobs/{job_key}.{format}

This call takes a job key or recno and returns detailed profile information about the job. This method returns an exhaustive list of attributes associated with the job.

DEFINITION

```
profile.getSpecific(key, callback);
```

EXAMPLE REQUEST

```
var Profile = require('upwork-
api/lib/routers/jobs/profile.js').Profile
;
```

Required key permissions

No special permissions are required

Arguments

job_key: **required, string**

The job profile key or a list of keys, separated by semicolon (';'), number of keys per request is limited to 20.

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Error messages

400: Bad Request

Possible output fields

(Click here to show/hide)

Returns

This resource returns an exhaustive list of attributes about the job. We recommend to pull data for a few freelancers and match up the fields to understand the meaning of the fields. Note that most of the response fields are human readable.

```
var profile = new Profile(api);
profile.getSpecific('~abcdf12345',
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE


```
{'amount': '5',
 'assignment_info': '',
 'assignments': {'assignment':
[{'as_ciphertext': '~abcdf1234567',
    'as_ciphertext_opening_recno':
'~abcdf1234549353a08',
    'as_engagement_title': 'API test
job',
    'as_from': '02/2014',
    'as_job_type': 'Fixed',
    'as_opening_title': 'API test job',
    'as_rate': '$0.00',
    'as_status': 'Active',
    'as_to': 'Present',
    'as_total_charge': '0',
    'as_total_hours': '0'},
   {'as_ciphertext': '~abcdf1234567',
    'as_ciphertext_opening_recno':
'~abcdf123454e8f073e',
    'as_engagement_title': 'Testing an
API',
    'as_from': '10/2013',
    'as_job_type': 'Fixed',
    'as_opening_title': 'Test create job
via API',
    'as_rate': '$0.00',
    'as_status': 'Closed',
    'as_to': '10/2013',
    'as_total_charge': '1.11',
    'as_total_hours': '0',
    'feedback': {'comment': 'Test Company
is a very good client, clean specs and
nice to communicate.',
     'comment_is_public': '1',
```

```
          'score': '5.00',
          'scores': {'score': [{'description':
    'competency and skills for the job,
    understanding of task complexities',
            'label': 'Skills',
            'score': '5'},
           {'description': 'quality of
    specifications/instructions',
            'label': 'Quality',
            'score': '5'},
           {'description': 'online presence
    on a consistent schedule',
            'label': 'Availability',
            'score': '5'},
           {'description': 'understanding of
    complexities and trade-offs',
            'label': 'Deadlines',
            'score': '5'},
           {'description': 'communication
    skills and responsiveness, feedback and
    guidance',
            'label': 'Communication',
            'score': '5'},
           {'description': 'cooperation and
    flexibility, open to suggestions for
    improvement',
            'label': 'Cooperation',
            'score': '5'}]}}}]},
     'buyer': {'op_adjusted_score': '5',
      'op_city': 'Bangkok',
      'op_contract_date': 'August 25, 2013',
      'op_country': 'Thailand',
      'op_timezone': 'UTC-08:00 Pacific Time
    (US & Canada); Los Angeles',
      'op_tot_asgs': '2',
      'op_tot_charge': '1.11',
      'op_tot_fp_asgs': '2',
      'op_tot_hours': '0',
      'op_tot_jobs_filled': '2',
      'op_tot_jobs_open': '0',
      'op_tot_jobs_posted': '5'},
     'candidates': '',
     'ciphertext': '~abcdf1234542a395a4',
     'engagement_weeks': '',
```

```
 'interviewees_total_active': '0',
 'job_category_level_one': 'Web, Mobile &
Software Dev',
 'job_category_level_two': 'Web
Development',
 'job_type': 'Fixed',
 'op_attached_doc': '',
 'op_cny_upm_verified': '1',
 'op_contractor_tier': '',
 'op_ctime': '1377423220000',
 'op_description': 'Will be discussed.',
 'op_engagement': 'Full-time - 30+
hrs/week',
 'op_high_hourly_rate_all': '0',
 'op_low_hourly_rate_all': '0',
 'op_other_jobs': '',
 'op_pref_english_skill': '0',
 'op_pref_fb_score': '0',
 'op_pref_has_portfolio': '0',
 'op_pref_hourly_rate_max': '',
 'op_pref_hourly_rate_min': '',
 'op_pref_location': '',
 'op_pref_odesk_hours': '0',
 'op_required_skills': '',
 'op_title': 'Test Upwork API',
 'op_tot_cand': '0',
 'op_tot_feedback': '1',
 'op_pref_group_id': 'somegroup',
 'op_pref_group_name': 'Super group',
 'op_pref_group_logo': 'http://...',
 'ui_opening_status': 'Closed'}
```

Invite freelancer to an interview

Endpoint

POST /api/hr/v1/jobs/{job_key}/candidates.{format}

This call can be used to invite a selected freelancer to an

DEFINITION

```
interviews.invite(jobKey, params,
callback);
```

interview for the job referenced.

Required key permissions

Create, modify and remove job posts

Arguments

job_key:	**required, string** Job key. For example: `~0150f0d859bb3453d0`.
format:	**optional, string** **Default:** `xml` **Valid values:** `json, xml` Response format.

Parameters

profile_key:	**conditionally required, string** The unique freelancer's key. Example: `~~677961dcd7f65c01`. Either `profile_key` or `provider__reference` parameter should be specified.
provider__refere nce:	**conditionally required, string** The freelancer's reference ID.Example: `12345`. It is used if the `profile_key` param is absent.
cover:	**required, string** The text of the cover letter.

Error messages

403:	Forbidden

EXAMPLE REQUEST

```
var Interviews = require('upwork-
api/lib/routers/hr/interviews.js').Interv
iews;

var interviews = new Interviews(api);
var jobKey = '~aaabb99900';
var params = {
  'profile_key': '~abcdf12345',
  'cover': 'Please accept this interview'
};
interviews.invite(jobKey, params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{'auth_user': {'first_name': 'John',
               'last_name': 'Johnson',
               'timezone': 'Asia/Omsk',
               'timezone_offset':
'25200'},
 'code': '200',
 'job_info': {'company_id': ''},
 'message': 'OK',
 'server_time': '1404389611'}
```

Returns

If successful, this resource returns a `200 OK` message.

Companies and teams

This section describes resources that allow you to manage companies and teams on Upwork.

List companies

Endpoint

GET /api/hr/v2/companies.{format}

This call returns all the companies the currently authorized user has access to on Upwork. It is important to take into account that just because a user has access to a company does not mean he/she has access to all areas of a company. To find out exactly what permissions a user has within a specific company use 'Get roles' API call.

Required key permissions

View the structure of your companies/teams

Arguments

format:	**optional, string**
	Default: `xml`
	Valid values: `json, xml`

DEFINITION

```
companies.getList(callback);
```

EXAMPLE REQUEST

```
var Companies = require('upwork-
api/lib/routers/organization/companies.js
').Companies;

var companies = new Companies(api);
companies.getList(function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
[
  {'name': 'A Test Company',
   'owner_ciphertext': '~~abcdf12345',
   'payment_verification_status':
'VERIFIED',
   'reference': '12345'
```

Response format.

Error messages

401:	Unauthorized
403:	Forbidden

Possible output fields

(Click here to show/hide)

Returns

This resource returns the list of companies that the currently authorized user has access to.

Get company

Endpoint

GET /api/hr/v2/companies/{company_reference}.{format}

Returns details regarding a specific company. If a user does not have access to this company, the call returns a 403 error. This API call does not return a list of teams within the company.

Required key permissions

View the structure of your companies/teams

Arguments

company_referen ce:	**required, string** The reference ID of the company.
format:	**optional, string**

```
    },
    {
      # Another team
    },
    # ...
]
```

DEFINITION

```
companies.getSpecific(cmpReference,
callback);
```

EXAMPLE REQUEST

```
var Companies = require('upwork-
api/lib/routers/organization/companies.js
').Companies;

var companies = new Companies(api);
companies.getSpecific('12345',
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{'name': 'A Test Company',
```

Default: `xml`

Valid values: `json, xml`

Response format.

Error messages

401: Unauthorized

403: Forbidden

Possible output fields

(Click here to show/hide)

Returns

This resource returns the details of the referenced company. If a user does not have access to the company, the resource returns a 403 error. This resource does not return the list of teams within the company.

List teams in company

Endpoint

GET

/api/hr/v2/companies/{company_reference}/teams.{format}

Returns a list of teams within the company being referenced (as long as the user has access to the referenced company).

Required key permissions

View the structure of your companies/teams

Arguments

```
    'owner_ciphertext': '~~abcdf12345',
    'payment_verification_status':
  'VERIFIED',
    'reference': '12345'}
```

DEFINITION

```
companies.getTeams(cmpReference,
callback);
```

EXAMPLE REQUEST

```
var Companies = require('upwork-
api/lib/routers/organization/companies.js
').Companies;

var companies = new Companies(api);
companies.getTeams('12345',
function(error, data) {
  console.log(data);
});
```

company_referen ce:	**required, integer** The reference ID of the company. It can be found using `List companies` API call.
format:	**optional, string** **Default:** `xml` **Valid values:** `json, xml` Response format.

Error messages

401:	Unauthorized
403:	Forbidden

Possible output fields

(Click here to show/hide)

Returns

This resource returns the list of teams in the referenced company.

List users in company

Endpoint

GET

/api/hr/v2/companies/{company_reference}/users.{format}

Returns a list of all users within the referenced company (only available for users with hiring privileges for the company called). These users may be in different teams, so if you want to see users grouped by teams use the 'List

EXAMPLE RESPONSE

```
[
  {'company__reference': '377329',
   'company_name': 'A Test Company',
   'id': 'abcdfghijk',
   'name': 'A Test Company',
   'parent_team__id': 'abcdfghijk',
   'parent_team__name': 'A Test Company',
   'parent_team__reference': '12345',
   'payment_verification_status':
'VERIFIED',
   'reference': '12345'
  },
  {
    # Another team
  },
  # ...
]
```

DEFINITION

```
companies.getUsers(cmpReference,
callback);
```

EXAMPLE REQUEST

```
var Companies = require('upwork-
api/lib/routers/organization/companies.js
').Companies;
```

team's users' API call'.

Required key permissions

View the structure of your companies/teams

Arguments

company_referen ce:	**required, integer**	The reference ID of the company. It can be found using `List companies` API call.
format:	**optional, string**	**Default:** `xml` **Valid values:** `json, xml` Response format.

Error messages

401:	Unauthorized
403:	Forbidden

Possible output fields

(Click here to show/hide)

Returns

This resource returns the list of users within the referenced company.

List teams

Endpoint

GET /api/hr/v2/teams.{format}

```
var companies = new Companies(api);
companies.getUsers('12345',
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
[
  {
    'first_name': 'John',
    'has_contract': '0',
    'id': 'john_johnson',
    'is_provider': '1',
    'last_name': 'Johnson',
    'public_url': 'https://...',
    'reference': '12345',
    'status': 'active',
    'timezone_offset': '25200'
  },
  {
    # Another teammate
  },
  # ...
]
```

DEFINITION

GET /api/hr/v2/teams.{format}

This call returns all the teams that a user has access to. Keep in mind that it returns teams across different companies.

Required key permissions

View the structure of your companies/teams

Arguments

format:	**optional, string**
	Default: `xml`
	Valid values: `json, xml`
	Response format.

Error messages

401:	Unauthorized
403:	Forbidden

Possible output fields

(Click here to show/hide)

Returns

This resource returns the list of all teams that the currently authorized user has access to.

List users in team

Endpoint

GET /api/hr/v2/teams/{team_reference}/users.{format}

```
teams.getList(callback);
```

EXAMPLE REQUEST

```
var Teams = require('upwork-
api/lib/routers/organization/teams.js').T
eams;

var teams = new Teams(api);
teams.getList(function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
[
  {
    'company__reference': '12345',
    'company_name': 'A Test Company',
    'id': 'abcdfghijk',
    'name': 'A Test Team',
    'parent_team__id': 'abcdfghijk',
    'parent_team__name': 'A Test
Company',
    'parent_team__reference': '12345',
    'reference': '12345'
   },
   {
     # Another team
   },
   # ...
 ]
```

DEFINITION

GET /api/hr/v2/teams/{team_reference}/users.{format}

This call returns a list of teams within the company being referenced (as long as the user has access to the referenced company).

Required key permissions

View the structure of your companies/teams

Arguments

team_reference:	**required, integer** The reference ID of the team.
format:	**optional, string** **Default:** `xml` **Valid values:** `json, xml` Response format.

Error messages

401:	Unauthorized
403:	Forbidden

Possible output fields

(Click here to show/hide)

Returns

This resource returns the following details of all the users in the referenced team:

reference:	Reference ID of the team.

status

```
teams.getUsersInTeam(teamReference,
callback);
```

EXAMPLE REQUEST

```
var Teams = require('upwork-
api/lib/routers/organization/teams.js').T
eams;

var teams = new Teams(api);
teams.getUsersInTeam('12345',
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
[
  {
    'first_name': 'John',
    'has_contract': '0',
    'id': 'john_johnson',
    'is_provider': '1',
    'last_name': 'Johnson',
    'public_url': 'https://...',
    'reference': '12345',
    'status': 'active',
    'timezone_offset': '25200'
  },
  {
    # Another teammate
  },
  # ...
]
```

timezone_offset: The user's timezone.

id: The user ID.

is_provider: Indicates whether a user is a freelancer. The value *1* means "yes" and *0* means "no".

last_name: The user's last name.

first_name: The user's first name.

List team rooms

Endpoint

GET /api/team/v2/teamrooms.{format}

This call retrieves all teamrooms the currently authenticated user has access to.

Required key permissions

View your workdiary

Arguments

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Error messages

401: Unauthorized

403: Forbidden

DEFINITION

```
teams.getList(callback);
```

EXAMPLE REQUEST

```
var Teams = require('upwork-
api/lib/routers/teams.js').Teams;

var teams = new Teams(api);
teams.getList(function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
[
  {
    'company_name': 'A Test Company',
    'id': 'abcdefghijk',
    'name': 'A Test Team',
    'parent_team_ref': '12345',
    'team_ref': '12345',
```

Returns

This resource returns the following details of the teams accessible to the currently authenticated user:

teamrooms:	A container for team rooms.
teamroom:	A container for team room items.
company_recno:	A unique ID number of the current company.
company_name:	The name of the current team company.
name:	The name of the current team.
id:	The ID name of the current team.
recno:	A unique ID number of the current team.
teamroom_api:	The current teamroom resource.

Get team room

Endpoint

GET /api/team/v2/teamrooms/{company_or_team}.{format}

This resource retrieves a team room for the company or team referenced.

Required key permissions

View your workdiary

```
    'teamroom_api':
'/api/team/v2/teamrooms/abcdefghijk.json'
  },
  {
    # Another teamroom
  },
  # ...
]
```

DEFINITION

```
teams.getSpecific(team, params,
callback);
```

EXAMPLE REQUEST

```
var Teams = require('upwork-
api/lib/routers/teams.js').Teams;
```

Arguments

company_or_tea m:	**required, string** The company ID or team ID. Use `List companies` or `List teams` API calls to get the IDs.
format:	**optional, string** **Default:** `xml` **Valid values:** `json, xml` Response format.

Parameters

online:	**optional, string** **Default:** `now` **Valid values:** `now, last_24h, all` Filters freelancers by work hours. Values description: `now` - Retrieves details for all currently logged in team members; `last_24h` - Retrieves details for all team members logged in during the past hour; `all` - Retrieves details for all team members.
disabled:	**optional, string** **Valid values:** `yes, no` If set to `yes`, disabled users are returned in the response.

Error messages

401:	Unauthorized

```
var teams = new Teams(api);
var team = 'abcdf12345';
var params = {};
teams.getSpecific(team, params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE


```
[
  {
    'account_status': 'enabled',
    'active_window_title': 'Upwork Team
Client',
    'activity': '5',
    'billing_status': 'billed.active',
    'cellts': '1404472378',
    'client_version': '',
    'company_id': 'abcdf12345',
    'computer_name': '',
    'digest': '',
    'keyboard_events_count': '223',
    'last_worked': '1404472378',
    'last_worked_snapshot_api':
'/api/team/v1/snapshots...',
    'last_worked_status': 'now',
    'memo': 'New Upwork Public API
documentation',
    'mouse_events_count': '0',
    'online_presence':
'13,6,6,0,26,73,33,20',
    'online_presence_img': 'https://...',
    'portrait_50_img': 'https://...',
    'portrait_img': 'https://...',
    'profile_url': 'https://...',
    'report24_img': 'https://...',
    'report_url': 'https://...',
    'role': 'contractor',
    'screenshot_img': 'https://...',
    'screenshot_img_lrg': 'https://...',
    'screenshot_img_med': 'https://...',
    'screenshot_img_thmb': 'https://...',
```

401: Unauthorized

403: Forbidden

Returns

This resource returns the following details of the snapshots:

Snapshot: This snapshot response is unique to the resource called. In this case, it refers to the team responses when using the team resource.

status: The current status of the snapshot.

time: Timestamp of the current snapshot taken.

billing_status: The current status of the user, *billed* or *active*.

report_url: URL linking to a detailed report of the user's activity.

activity: The level of the user's activity, on a scale of *0* (low) to *10* (high).

last_worked_status: The date time when a user last worked. If the user is online, the time is *now*.

memo: Current user memo as entered into the Upwork team client.

```
        'screenshot_url': 'https://...',
        'snapshot_api':
    '/api/team/v1/snapshots...',
        'status': 'NORMAL',
        'time': '1404472378',
        'timezone': 'UTC+07:00 Bangkok,
    Jakarta, Hanoi',
        'uid': '',
        'user': {'archiving_time':
    '1404086400',
        'creation_time': '1376941475',
        'first_name': 'John',
        'last_name': 'Johnson',
        'mail': 'john_johnson@example.com',
        'messenger_id': '',
        'messenger_type': '',
        'timezone': 'Asia/Bangkok',
        'timezone_offset': '25200',
        'uid': 'john_johnson'},
        'user_id': 'john_johnson',
        'view_assignment_url': 'https://...',
        'workdiary_api':
    '/api/team/v1/workdiaries...'}]
```

Get Work Diary

Endpoint

GET

/api/team/v1/workdiaries/{company_id}/{username}/{date}.{format}

The Work Diary method retrieves all snapshots from a single user account within a single day. Keep in mind that a day is dependent on server time and not the day in which the query is made. Make sure to test with various locations before you're done.

Required key permissions

View your workdiary

Arguments

company_or_team:	**required, string** The company ID or team ID. Use 'List companies' or 'List teams' API calls to get the IDs.
username:	**required, string** The username of the target user account. This is your or your team mate username.
date:	**required, string** The target date in `yyyymmdd` format. If absent, it defaults to today.
format:	**optional, string**

DEFINITION

```
workdiary.get(
  company, username, date,
  params, callback);
```

EXAMPLE REQUEST

```
var Workdiary = require('upwork-
api/lib/routers/workdiary.js').Workdiary;

var workdiary = new Workdiary(api);
var params = {};
workdiary.get('abcde12345',
'john_johnson', '20140704', params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
(
  {
    'archiving_time': '1404691200',
    'creation_time': '1376941475',
    'first_name': 'John',
    'last_name': 'Johnson',
    'mail': 'john_johnson@example.com',
    'status': 'enabled',
    'timezone': 'Asia/Bangkok',
    'timezone_offset': '25200',
    'uid': 'john_johnson'
  },
  [
    {
      'account_status': '',
      'active_window_title': 'emacs@user-
laptop',
```

format: optional, string

Default: `xml`

Valid values: `json, xml`

Response format.

Parameters

tz: optional, string

Default: `mine`

Valid values: `mine, user, gmt`

The time zone to use.

Error messages

401:	Unauthorized
403:	Forbidden (user is not invited to specified company)

Returns

This resource returns the following details of the snapshots:

user:	The user of the Work diary.
timezone:	The time zone of the current authenticated user.
uid:	The user ID of the current authenticated user.
timezone_offset:	The difference between current time and GMT.
creation_time:	Time when the current Work diary entry was created. Format: UNIX timestamp.

```
        'activity': '2',
        'allow_delete': 'true',
        'allow_edit': 'true',
        'billing_status': 'billed.active',
        'cell_time': '1404735600',
        'cellts': '',
        'client_version': 'Linux/3.12.9',
        'company_id': 'abcdef12345',
        'computer_name': 'user-laptop',
        'digest': '',
        'keyboard_events_count': '142',
        'memo': 'New Upwork Public API
documentation',
        'mouse_events_count': '33',
        'screenshot_img': 'https://...',
        'screenshot_img_lrg':
'https://...',
        'screenshot_img_med':
'https://...',
        'screenshot_img_thmb':
'https://...',
        'screenshot_url': 'https://...',
        'snapshot_api':
'/api/team/v1/snapshots/...',
        'status': 'NORMAL',
        'time': '1404736072',
        'timezone': '',
        'uid': 'abcdff38-9999-4444-9999-
adcdfegihgklm',
        'user_id': 'john_johnson'
      },
      {
        # Another entry
      },
      # ...
    ]
)
```

mail: The default mail ID of the current authenticated user.

last_name: The user's last name.

first_name: The user's first name.

snapshot: The snapshot details.

cell_time: The time when the snapshot was captured.

allow_edit: Defines if the current entry is editable or not.

status: The status of the current entry.

time: The timestamp of the current entry.

billing_status: Billing status.

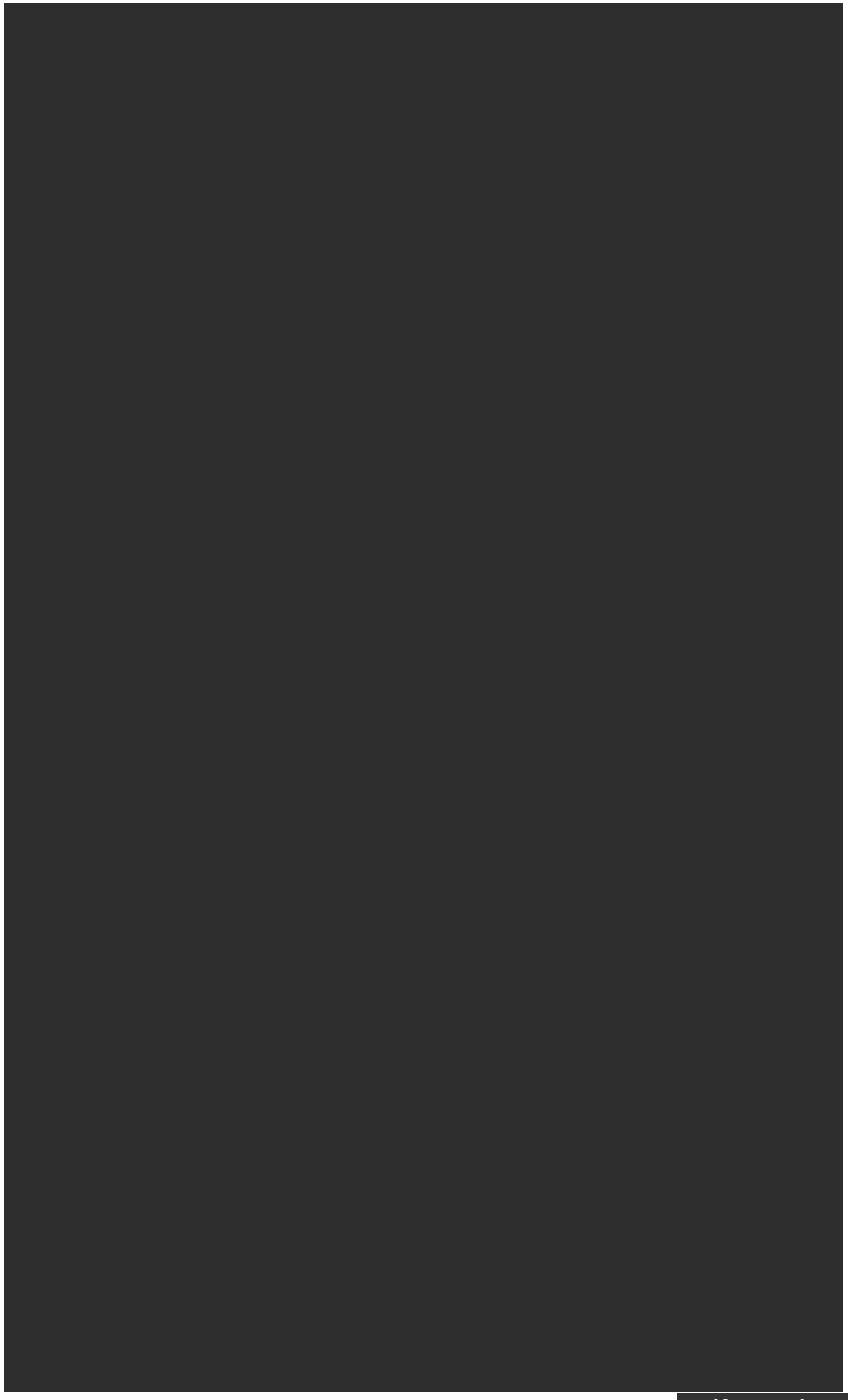
screenshot_img: URL to the screenshot of the *Work diary* image.

activity: The level of the user's activity; on a scale of `0` (low) to `10` (high).

screenshot_url: The URL of the screenshot for this entry.

mouse_events_count: The number of mouse events associated with this entry.

company_id: The Upwork ID of the company associated with the user in this entry.



timezone: The time zone of the entry.

uid: The user ID of the user.

keyboard_events_count: The number of keyboard events associated with this entry.

memo: The memo associated with the current entry.

active_window_title: The name of the active window at the user's device when the entry was made.

computer_name: The name of the user's device during the current entry.

screenshot_img_thmb: URL of a thumbnail of the current screenshot.

user_id: The user's Upwork ID.

client_version: The client version used.

snapshot_api: The resource used for the snapshot.

Get Work Diary by Contract

Endpoint

GET

/api/team/v2/workdiaries/contracts/{contract_id}/{date}.{format}

DEFINITION

```
workdiary.getByContract(
  contract_id, date,
  params, callback);
```

The Work Diary method retrieves all snapshots from a single user account within a single day. Keep in mind that a day is dependent on server time and not the day in which the query is made. Make sure to test with various locations before you're done.'

Required key permissions

View your workdiary

Arguments

contract_id: **required, string**

The contract ID or a list of contract IDs, separated by semicolon (';').

date: **required, string**

The target date in `yyyymmdd` format. If absent, it defaults to today.

format: **optional, string**

Default: `json`

Valid values: `json`

Response format.

Parameters

tz: **optional, string**

Default: `mine`

Valid values: `mine, user, gmt`

The time zone to use.

Error messages

401: Unauthorized

403: Forbidden

EXAMPLE REQUEST

```
var Workdiary = require('upwork-
api/lib/routers/workdiary.js').Workdiary;

var workdiary = new Workdiary(api);
var params = {};
workdiary.getByContract('12345;78912',
'20140704', params, function(error, data)
{
  console.log(data);
});
```

EXAMPLE RESPONSE

```
(
  {
    'archiving_time': '1404691200',
    'creation_time': '1376941475',
    'first_name': 'John',
    'last_name': 'Johnson',
    'mail': 'john_johnson@example.com',
    'status': 'enabled',
    'timezone': 'Asia/Bangkok',
    'timezone_offset': '25200',
    'uid': 'john_johnson'
  },
  [
    {
      'account_status': '',
      'active_window_title': 'emacs@user-
laptop',
      'activity': '2',
      'allow_delete': 'true',
      'allow_edit': 'true',
      'billing_status': 'billed.active',
      'cell_time': '1404735600',
      'cellts': '',
      'client_version': 'Linux/3.12.9',
      'company_id': 'abcdef12345',
      'computer_name': 'user-laptop',
      'digest': '',
      'keyboard_events_count': '142',
```

400: Bad Request

Returns

This resource returns the following details of the snapshots:

user: The user of the Work diary.

timezone: The time zone of the current authenticated user.

uid: The user ID of the current authenticated user.

timezone_offset: The difference between current time and GMT.

creation_time: Time when the current Work diary entry was created. Format: UNIX timestamp.

mail: The default mail ID of the current authenticated user.

last_name: The user's last name.

first_name: The user's first name.

snapshot: The snapshot details.

cell_time: The time when the snapshot was captured.

allow_edit: Defines if the current entry is editable or not.

status: The status of the current entry.

```
        'memo': 'New Upwork Public API
    documentation',
        'mouse_events_count': '33',
        'screenshot_img': 'https://...',
        'screenshot_img_lrg':
    'https://...',
        'screenshot_img_med':
    'https://...',
        'screenshot_img_thmb':
    'https://...',
        'screenshot_url': 'https://...',
        'snapshot_api':
    '/api/team/v1/snapshots/...',
        'status': 'NORMAL',
        'time': '1404736072',
        'timezone': '',
        'uid': 'abcdff38-9999-4444-9999-
    adcdfegihgklm',
        'user_id': 'john_johnson'
      },
      {
        # Another entry
      },
      # ...
    ]
)
```

time: The timestamp of the current entry.

billing_status: Billing status.

screenshot_img: URL to the screenshot of the *Work diary* image.

activity: The level of the user's activity; on a scale of `0` (low) to `10` (high).

screenshot_url: The URL of the screenshot for this entry.

mouse_events_count: The number of mouse events associated with this entry.

company_id: The Upwork ID of the company associated with the user in this entry.

timezone: The time zone of the entry.

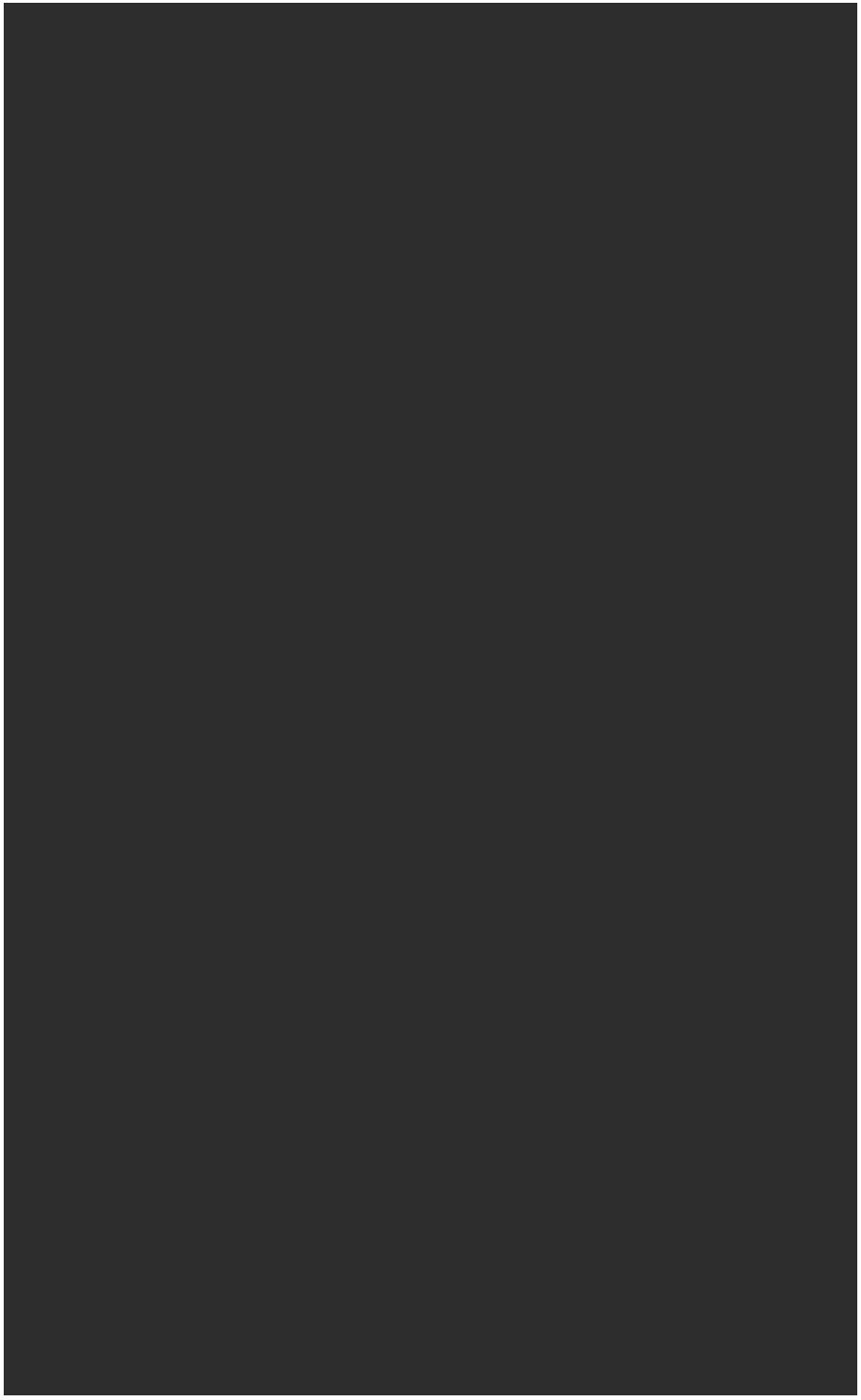
uid: The user ID of the user.

keyboard_events_count: The number of keyboard events associated with this entry.

memo: The memo associated with the current entry.

active_window_title: The name of the active window at the user's device when the entry was made.

computer_name: The name of the user's device



during the current entry.

screenshot_img_thmb: URL of a thumbnail of the current screenshot.

user_id: The user's Upwork ID.

client_version: The client version used.

snapshot_api: The resource used for the snapshot.

Get Workdays by Company

Endpoint

GET

/api/team/v2/workdays/companies/{company_id}/{from_date,till_date}.{format}

The Workdays method retrieves all work days within a specific period and in a specific company for curently authenticated user. Keep in mind that a day is dependent on server time and not the day in which the query is made. Make sure to test with various locations before you're done.'

Required key permissions

View your workdiary

Arguments

company_id: **required, string**

The company ID.

DEFINITION

```
workdays.getByCompany(
  company_id, from_date, till_date,
  params, callback);
```

EXAMPLE REQUEST

```
var Workdays = require('upwork-
api/lib/routers/workdays.js').Workdays;

var workdays = new Workdays(api);
var params = {};
workdays.getByCompany('testcompany',
'20150704', '20150710', params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
(
  {
```

from_date: **required, string**

The start date in `yyyymmdd` format.

end_date: **required, string**

The end date in `yyyymmdd` format.

format: **optional, string**

Default: `json`

Valid values: `json`

Response format.

Parameters

tz: **optional, string**

Default: `mine`

Valid values: `mine, user, gmt`

The time zone to use.

Error messages

401: Unauthorized

403: Forbidden

400: Bad Request

Returns

This resource returns the list of active workdays in the company for the currently authenticated user, and the following details:

range: Specified range.

workday.date: List of workdays.

workday.workdiary_api: The resource used for Workdiary.

```
      'archiving_time': '1404691200',
      'creation_time': '1376941475',
      'first_name': 'John',
      'last_name': 'Johnson',
      'mail': 'john_johnson@example.com',
      'status': 'enabled',
      'timezone': 'Asia/Bangkok',
      'timezone_offset': '25200',
      'uid': 'john_johnson'
    },
    'workdays': {
      'range': {
        'to': '1288915200',
        'from': '1281052800'
      },
      'workday': {
        'date': [
          '20100816',
          '20100904'
        ],
        'workdiary_api':
'/api/team/v2/workdiaries/companies/testc
ompany.json'
      }
    }
)
```

Get Workdays by Contract

Endpoint

GET

/api/team/v2/workdays/contracts/{contract_id}/{from_date,till_date}.{format}

The Workdays method retrieves all work days within a specific period by contract. Keep in mind that a day is dependent on server time and not the day in which the query is made. Make sure to test with various locations before you're done.'

Required key permissions

View your workdiary

Arguments

contract_id: **required, string**

The contract ID or a list of contract IDs, separated by semicolon (';').

from_date: **required, string**

The start date in `yyyymmdd` format.

end_date: **required, string**

The end date in `yyyymmdd` format.

format: **optional, string**

Default: `json`

Valid values: `json`

Response format.

Parameters

DEFINITION

```
workdays.getByContract(
  contract_id, from_date, till_date,
  params, callback);
```

EXAMPLE REQUEST

```
var Workdays = require('upwork-
api/lib/routers/workdays.js').Workdays;

var workdays = new Workdays(api);
var params = {};
workdays.getByContract('12345',
'20150704', '20150710', params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
(
  {
    'archiving_time': '1404691200',
    'creation_time': '1376941475',
    'first_name': 'John',
    'last_name': 'Johnson',
    'mail': 'john_johnson@example.com',
    'status': 'enabled',
    'timezone': 'Asia/Bangkok',
    'timezone_offset': '25200',
    'uid': 'john_johnson'
  },
  'workdays': {
    'range': {
      'to': '1288915200',
      'from': '1281052800'
    },
    'workday': {
```

tz: optional, string

Default: `mine`

Valid values: `mine, user, gmt`

The time zone to use.

Error messages

401:	Unauthorized
403:	Forbidden
400:	Bad Request

Returns

This resource returns the list of active workdays and the following details:

range:	Specified range.
workday.date:	List of workdays.
workday.workdiary_api:	The resource used for Workdiary.

```
        'date': [
          '20100816',
          '20100904'
        ],
        'workdiary_api':
'/api/team/v2/workdiaries/contracts/12345
.json'
      }
  }
)
```

DEFINITION

Contracts and offers

This section describes resources that allow you to manage contracts and offers. For example, you can send offers, create and close contracts and manage existing engagements.

List client offers

Endpoint

GET /api/offers/v1/clients/offers.{format}

This resource lists all offers made by a client.

Required key permissions

View your job offers

Arguments

format: optional, string

Default: `json`

Valid values: `json, xml`

Response format.

Parameters

company_id: conditionally required, string

The client's company reference ID. Example: `34567`. Get it using 'List companies' API call. If `company_id` is not specified, the API will infer it from job's data relying on the provided `job__reference` parameter. Either `company_id` or `job__reference` parameter must be specified.

team__reference: conditionally required, string

The client's team reference ID. Example: `34567`. Get it using 'List teams' API call. If `team__reference` is not specified, the API will infer it from job's data relying on the provided `job__reference`. Note that either

```
offers.getList(params, callback);
```

EXAMPLE REQUEST

```
var Offers = require('upwork-
api/lib/routers/hr/clients/offers.js').Of
fers;

var offers = new Offers(api);
var params = {'company_id': '12345'};
offers.getList(params, function(error,
data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
 {'auth_user': {'first_name': 'John',
                'last_name': 'Johnson',
                'timezone': 'Asia/Omsk',
                'timezone_offset':
'25200'},
  'offers': {
    'lister': {'paging': {'count': '20',
'offset': '0'},
               'query': '',
               'sort': '',
               'total_items': ''},
    'offer': [
      {
        'client_org_ref': '55102',
        'client_user_ref': '106975',
       'contractor_org_ref': '',
        'contractor_user_ref': '3',
        'is_visible_to_contractor': '',
        'job_posting_ref': '',
        'last_event_state': 'new',
        'rid': '70727',
        'terms_data': {
          'charge_rate': '21.0',
          'charge_weekly_stipend_amount':
```

`job__reference`. Note that either `job__reference` or `team__reference` parameters should be specified.

job__reference: **conditionally required, string**

The job reference ID. Get it using 'List jobs' API call. Note that either `job__reference` or `team__reference` parameters should be specified.

status: **optional, string**

Default: `new`

Valid values: `accepted, new, declined, expired, withdrawn, cancelled, changed`

The current status of the offer. By default only offers in status `new` are returned.

page: **optional, string**

Pagination, formed as `$offset;$count`. Example: `page=20;10`

Error messages

401:	Unauthorized
403:	No recruiter or hiring manager permissions
403:	No access to team/company

Possible output fields

(Click here to show/hide)

Returns

```
    '',
              'manual_time_allowed': '',
              'start_date': '1405036800000',
              'type': 'hourly',
              'weekly_limit': ''
            },
            'title': 'Test offers API',
            'type': 'hourly'
          },
          {
            # Another offer object
          },
          # ..
        ]
      },
    'server_time': '1405046576'}
```

See “Possible output fields” section.

Get client offer

Endpoint

GET /api/offers/v1/clients/offers/{offer_id}.{format}

This call retrieves details about a specific offer as a client.

Required key permissions

View your job offers

Arguments

offer_id: **required, string**

Offer reference ID.

format: **optional, string**

Default: `json`

Valid values: `json, xml`

Response format.

Parameters

company_id: **conditionally required, string**

The client's company reference ID. Example: `34567`. Get it using 'List companies' API call. If `company_id` is not specified, the API infers it from job's data relying on the provided `job__reference` parameter. Either

DEFINITION

```
offers.getSpecific(reference, params,
callback);
```

EXAMPLE REQUEST

```
var offers = require('upwork-
api/lib/routers/hr/clients/offers.js').Of
fers;

var Offers = new Offers(api);
var reference = '12345';
var params = {'job__reference':
'~012345abcdef'};
offers.getSpecific(reference, params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {'first_name': 'John',
                'last_name': 'Johnson',
                'timezone': 'Asia/Omsk',
                'timezone_offset':
'25200'},
  'offer': {
    'client_org_ref': '12345',
    'client_user_ref': '102345',
    'contractor_org_ref': '',
    'contractor_user_ref': '102345',
```

`company_id` or `job__reference` parameter should be specified.

job__reference: **conditionally required, string**

The job reference ID. Get it using 'List jobs' API call. Either `job__reference` or `company_id` parameter should be specified.

Error messages

401:	Unauthorized
403:	No recruiter or hiring manager permissions
403:	No access to team/company

Possible output fields

(Click here to show/hide)

Returns

See “Possible output fields” section.

Send client offer

Endpoint

POST /api/offers/v1/clients/offers.{format}

This call sends an offer to a freelancer.

Required key permissions

Make a job offer on your behalf

Arguments

```
      'is_visible_to_contractor': '',
      'job_posting_ref': '',
      'last_event_state': 'new',
      'rid': '77777',
      'terms_data': {
        'charge_rate': '20.0',
        'charge_weekly_stipend_amount': '',
        'start_date': '1404777600000',
        'type': 'hourly',
        'weekly_limit': ''
      },
      'title': 'Test offers API send
  offer',
      'type': 'hourly'
    },
    'server_time': '1404906002'
  }
```

DEFINITION

```
offers.makeOffer(params, callback);
```

EXAMPLE REQUEST

```
var Offers = require('upwork-
api/lib/routers/hr/clients/offers.js').Of
fers;

var offers = new Offers(api);
```

format:	optional, string **Default:** `json` **Valid values:** `json, xml` Response format.

Parameters

title:	**required, string** The title of the offer/contract.
job_type:	**required, string** **Valid values:** `hourly, fixed-price` The type of the job.
charge_rate:	**required, number** The budget amount for fixed-price jobs or the hourly charge rate for hourly jobs.
message_to_contractor:	**required, string** Instructions and other job details for the freelancer.
related_jobcategory2:	**conditionally required, integer** Related job category (V2). Example: `531770282584862733`. You can specify a job category (V2) in three different ways: either by directly using the `related_jobcategory2` parameter, by using `context[related_jobcategory2]`, or by using `context[job_posting_ref]`. If

```
var params = {
  'title': 'Test offers API send offer
02',
  'job_type': 'hourly',
  'charge_rate': '20.0',
  'message_to_contractor': 'Please accept
this test job',
  'team__reference': '12345',
  'contractor_username':
'john_freelancer'
};
offers.makeOffer(params, function(error,
data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {'first_name': 'John',
                'last_name': 'Johnson',
                'timezone': 'Asia/Omsk',
                'timezone_offset':
'25200'},
   'offer_id': '12345',
   'server_time': 1404844979
 }
```

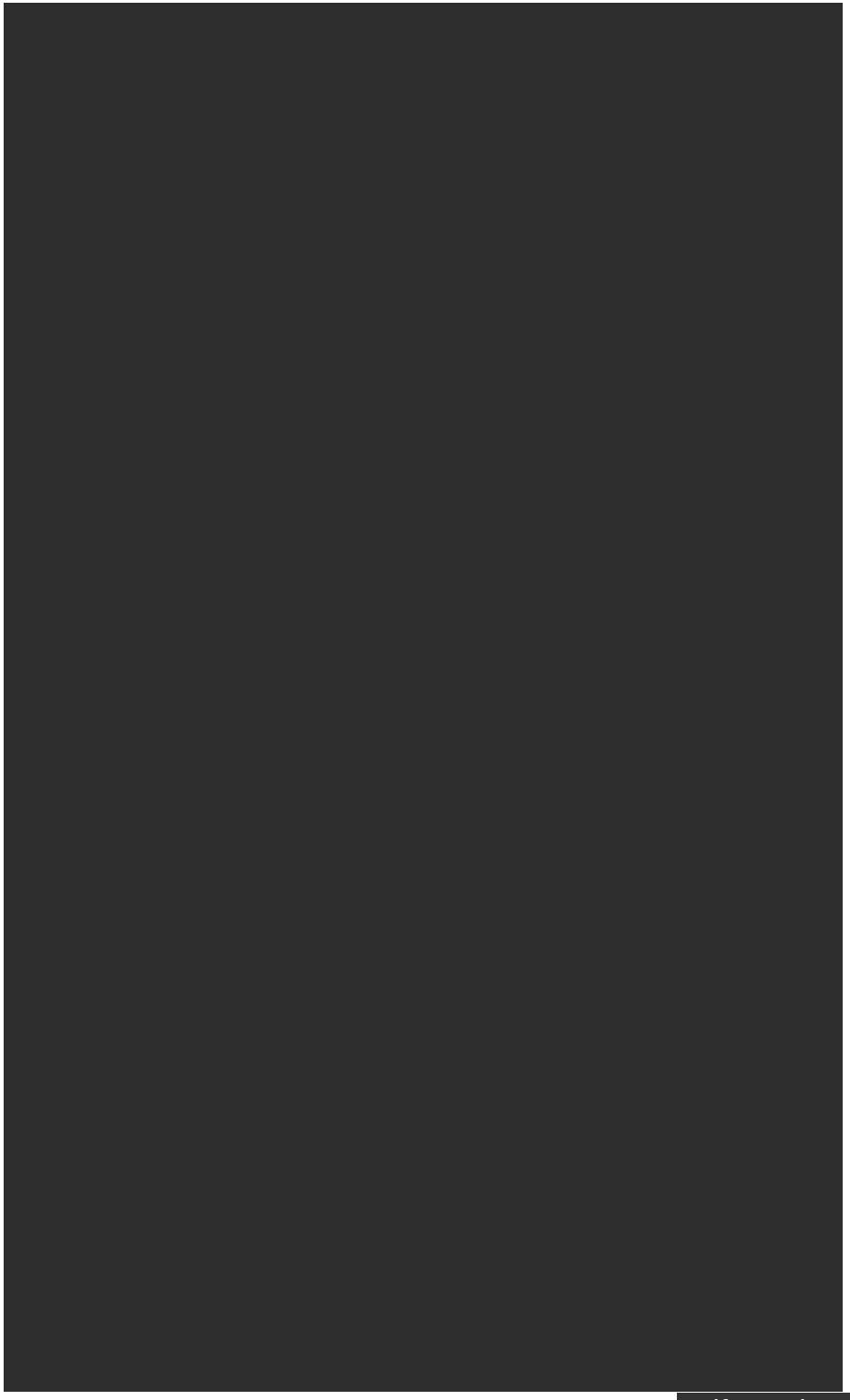
`context[job_posting_ref]` is specified, then the job category is read from the referenced job posting, and this parameter can be omitted. Get a list of available categories by calling the 'List categories V2' API.

team__reference: **conditionally required, string**

The client's team reference ID. Example: `34567`. This parameter is conditionally optional and required if you send offer 'on the fly' without providing context about any existing job. If you send offer for the existing job, the `team__reference` parameter becomes optional and will be taken automatically from job's data relying on the provided ``context[job_posting_ref]`` parameter. Use 'List teams' API call to get team reference ID.

client_team_ref: **conditionally required, string**

The client's team reference. Example: `mytestcompany:myteam`. This parameter is conditionally optional and required if you send offer 'on the fly' without providing context about any existing job. Use `List teams` API call to get it.



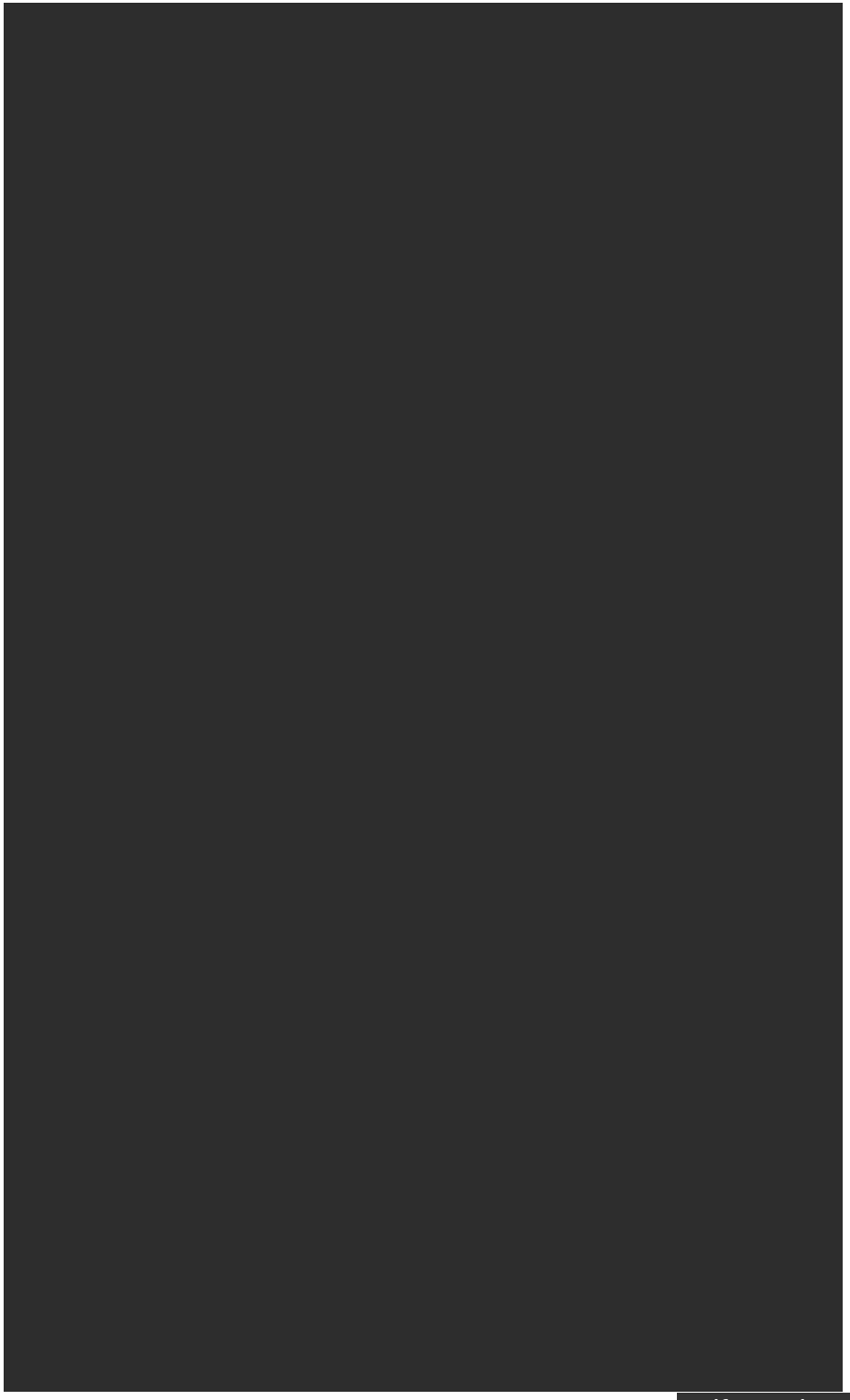
contractor_usern ame: **conditionally required, string**
The freelancer's username. Example: `contractoruid`. It can be ignored if `contractor_reference` or `contractor_key` parameter is set.

contractor_refere nce: **conditionally required, string**
The freelancer's reference ID. It is ignored if the parameter `contractor_username` or `contractor_key` is specified. Example: `1234`. You can use `Search freelancers` API call to get the freelancer's reference ID.

contractor_key: **conditionally required, string**
The unique profile key, used if `contractor_username` is absent. Example: `~~677961dcd7f65c01`.

contractor_org: **optional, string**
Optional param for cases of non exclusive agency contractors. Please provide here, freelancer's agency reference, to which the offer should be sent.

context: **conditionally required, string**
Additional data about the offer. This parameter is required in case you want to send an offer for the job that is already created. Valid array keys



are: `previous_offer_ref`, `job_posting_ref`, `job_application_ref`, `contract_ref`. Example: `context[job_posting_ref] = {{ opening_id }} & context[job_application_ref] = {{ application_id }}` where `job_posting_ref` is a job key, for example `~01c8e0xxxxxxxx05255`.

weekly_limit: **optional, integer**

The maximum number of hours per week the freelancer can bill for.

weekly_stipend: **optional, number**

An additional payment to be issued to the freelancer each week.

close_on_accept: **optional, integer**

Default: `1`

Valid values: `0, 1`

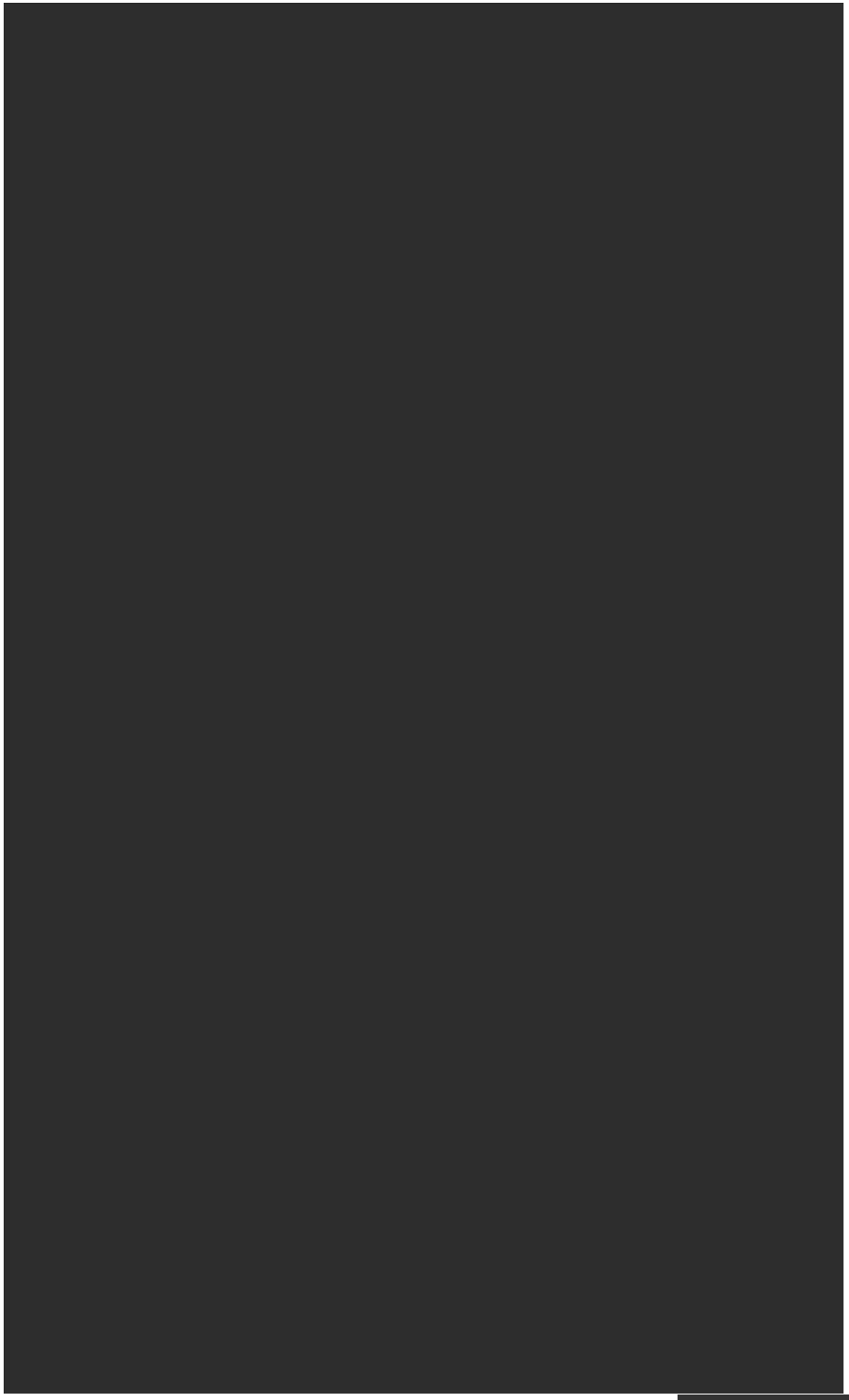
If the value is true, it automatically closes the related job post if this offer is accepted. The default value is true.

start_date: **optional, string**

The start date of the hourly-priced contract. Example: `06-15-2011`

milestones: **conditionally required, array**

Array of milestones for fixed-priced jobs in the following format: `milestones[0][$key]`, ...,



`milestones[N][$key]`, where key is one of the following - `milestone_description` (string), `deposit_amount` (float), `due_date` (string in format mm-dd-yyyy). Required for fixed-priced jobs.

Error messages

401:	Unauthorized
403:	No recruiter or hiring manager permissions
403:	No access to team/company
400:	Required field is missing

Returns

If the request is successful, a `200 OK` message is returned together with the offer ID.

List freelancer's offers

Endpoint

GET /api/offers/v1/contractors/offers.{format}

This call retrieves a list of offers received by a freelancer.

Required key permissions

View your job offers

Arguments

DEFINITION

```
offers.getList(params, callback);
```

EXAMPLE REQUEST

```
var Offers = require('upwork-
api/lib/routers/hr/freelancers/offers.js'
).Offers;

var offers = new Offers(api);
var params = {'status': 'new'};
```

format: optional, string

Default: `json`

Valid values: `json, xml`

Response format.

Parameters

status: optional, string

Default: `new`

Valid values: `accepted, new, declined, expired, withdrawn, cancelled, changed`

The current status of the offer. By default only offers with status `new` are returned.

page: optional, string

Pagination, formed as `$offset;$count`. Example: `page=20;10`.

Error messages

401: Unauthorized

403: Forbidden

Possible output fields

(Click here to show/hide)

Returns

See “Possible output fields” section.

```
offers.getList(params, function(error,
data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {'first_name': 'John',
                'last_name': 'Johnson',
                'timezone': 'Asia/Omsk',
                'timezone_offset':
'25200'},
  'offers': {
    'lister': {'paging': {'count': '20',
'offset': '0'},
               'query': '',
               'sort': '',
               'total_items': ''},
    'offer': [
      {
        'client_org_ref': '55102',
        'client_user_ref': '106975',
        'contractor_org_ref': '',
        'contractor_user_ref': '106976',
        'is_visible_to_contractor': '',
        'job_posting_ref': '',
        'last_event_state': 'new',
        'rid': '70449',
        'terms_data': {
          'charge_rate': '20.0',
          'charge_weekly_stipend_amount':
'',
          'manual_time_allowed': '',
          'start_date': '1404777600000',
          'type': 'hourly',
          'weekly_limit': ''
        },
        'title': 'Test offers API send
offer',
        'type': 'hourly'
      },
      {
```

Get freelancer's offer

Endpoint

GET /api/offers/v1/contractors/offers/{offer_id}.{format}

This call retrieves details about a specific offer received by a freelancer.

Required key permissions

View your job offers

Arguments

offer_id:	**required, string**
	Offer reference ID.
format:	**optional, string**
	Default: `json`
	Valid values: `json, xml`
	Response format.

Error messages

401:	Unauthorized
403:	Forbidden

Possible output fields

```
          # Another offer object
        },
        # ...
      ]
    },
    'server_time': '1405092898'
  }
```

DEFINITION

```
offers.getSpecific(reference, callback);
```

EXAMPLE REQUEST

```
var Offers = require('upwork-
api/lib/routers/hr/freelancers/offers.js'
).Offers;

var offers = new Offers(api);
var reference = '12345';
offers.getSpecific(reference,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{'auth_user': {'first_name': 'John',
               'last_name': 'Johnson',
               'timezone': 'Asia/Omsk',
               'timezone_offset':
'25200'},
 'offer': {
   'client_org_ref': '55102',
   'client_user_ref': '106975',
```

Possible output fields

(Click here to show/hide)

Returns

See “Possible output fields” section.

```
    'contractor_org_ref': '',
    'contractor_user_ref': '106976',
    'is_visible_to_contractor': '',
    'job_posting_ref': '',
    'last_event_state': 'new',
    'rid': '12345',
    'terms_data': {
      'charge_rate': '20.0',
      'charge_weekly_stipend_amount': '',
      'manual_time_allowed': '',
      'related_jobcategory2': '',
      'start_date': '1404777600000',
      'type': 'hourly',
      'version': 'v1',
      'weekly_limit': ''
    },
    'title': 'Test offers API resource
sends an offer',
    'type': 'hourly'
 },
 'server_time': '1405101130'}
```

Accept or decline an offer

Endpoint

POST /api/offers/v1/contractors/actions/{offer_id}.{format}

Allows a freelancer to accept or decline an offer. In order to use this call the authorized user needs offer owner or staffing manager permissions to the agency.

Required key permissions

Make a job offer on your behalf

Arguments

DEFINITION

```
offers.actions(params, callback);
```

EXAMPLE REQUEST

```
var Offers = require('upwork-
api/lib/routers/hr/freelancers/offers.js'
).Offers;

var offers = new Offers(api);
var offerId = '12345';
var params = {'name': 'accept'};
offers.actions(offerId, params,
function(error, data) {
```

offer_id: **required, string**

Offer reference ID.

format: **optional, string**

Default: `json`

Valid values: `json, xml`

Response format.

Parameters

name: **required, string**

Valid values: `accept, decline`

The name of the action.

Error messages

401: Unauthorized

403: Forbidden

400: Bad Request

Returns

This resource returns the following response message:

message: Result message.

List job applications as client

Endpoint

GET /api/hr/v3/clients/applications.{format}

This call lists all job applications received by a client.

Required key permissions

View your job applications

```
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {'first_name': 'John',
                'last_name': 'Johnson',
                'timezone': 'Asia/Omsk',
                'timezone_offset':
'25200'},
  'message': 'Offer 12345 accepted',
  'server_time': '1405092898'
}
```

DEFINITION

```
applications.getList(params, callback);
```

EXAMPLE REQUEST

```
var Applications = require('upwork-
api/lib/routers/hr/clients/applications.j
```

View your job applications

Arguments

format:	**optional, string** **Default:** `xml` **Valid values:** `json, xml` Response format.

Parameters

buyer_team__ref	**required, string**
erence:	The reference ID of the client's team. It allows getting applications for a specific team. Example: `34567`. Use 'List Teams' API call to get it.
job_key:	**required, string** The job key. It allows getting applications for a specific job. Example: `~01d54a7xxxxx125731`.
status:	**optional, string** **Valid values:** `shortlisted, messaged, hired, offered, declined, hidden` The current status of the job application.
profile_key:	**optional, string** Filters by a specific freelancer's profile key.
agency_team__re	**optional, string**
ference:	The reference ID of the agency.
order_by:	**optional, string**

```
s').Applications;

var applications = new Applications(api);
var params = {
  'buyer_team__reference': 12345,
  'job_key': '~12345abcdef'
};
applications.getList(params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE


```
{
  'applications': [
    {
      'contractor_ciphertext':
'~012345abcdef',
      'contractor_name': 'John Johnson',
      'contractor_portrait_url':
'https://...',
      'contractor_ref': '675604',
      'contractor_title': 'Software
developer - test WEBAPI oAuth',
      'cover_letter': '',
      'created_type': 'Client',
      'ctime_epoch': '1404794618',
      'engagement_duration_ref': '2',
      'feedback_score': '0',
      'fp_amount_agreed': '',
      'fp_pay_amount': '0',
      'fp_upfront_payment': '0',
      'hr_charge_rate': '22.22',
      'hr_pay_rate': '20',
      'interview_status': 'Waiting For
Provider',
      'invite_letter': 'Test invite to
interview via oAuth',
      'is_auto_hidden': 'f',
      'is_hidden_by_buyer': '0',
      'is_hidden_by_provider': '0',
      'is_seen_by_buyer': 't',
```

order_by: optional, string

Sorts results in format `$field_name1;$field_name2;..$field_nameN;AD...A`. Here `A` stands for ascending order, `D` - descending order. Example: `order_by=created_time;D`.

page: optional, string

Pagination, formed as `$offset;$count`. Example: `page=20;10`.

Error messages

401:	Unauthorized
403:	No recruiter or hiring manager permissions
403:	No access to team/company

Possible output fields

(Click here to show/hide)

Returns

See “Possible output fields” section.

```
        'is_shortlisted': '0',
        'is_undecided': '1',
        'job__description': 'Create job via
    oAuth api',
        'job_pref_matches': {'prefs_match':
    '0', 'prefs_total': '0'},
        'job_type': 'Hourly',
        'offer_status': '',
        'opening_ciphertext':
    '~012345abcdef',
        'rid': '12345',
        'status': 'In Process',
        'team_nid': 'unazcm8kj7ytdrwwgc-
    pkq',
        'total_hours': '0',
        'ui_opening_title': 'Test oAuth API
    create job'
      }
    ],
    'auth_user': {'first_name': 'John',
                  'last_name': 'Johnson',
                  'timezone': 'Asia/Omsk',
                  'timezone_offset':
    '25200'},
    'lister': {'paging': {'count': '10',
    'offset': '0'},
               'query': '',
               'sort': '',
               'total-count': '1',
               'total_items': ''},
    'server_time': '1405339447'
}
```

Get job application as client

Endpoint

GET /api/hr/v3/clients/applications/{application_id}.{format}

This call retrieves details about a specific job application

DEFINITION

```
applications.getSpecific(reference,
params);
```

received by a client.

Required key permissions

View your job applications

Arguments

application_id:	**required, string**
	The job application ID.
format:	**optional, string**
	Default: `xml`
	Valid values: `json, xml`
	Response format.

Parameters

buyer_team__reference:	**required, string**
	The reference ID of the client's team. It allows getting applications for a specific team. Example: `34567`. Use 'List Teams' API call to get it.

Error messages

401:	Unauthorized
403:	No recruiter or hiring manager permissions
403:	No access to team/company

Possible output fields

(Click here to show/hide)

Returns

See “Possible output fields” section.

EXAMPLE REQUEST

```
var Applications = require('upwork-
api/lib/routers/hr/clients/applications.j
s').Applications;

var applications = new Applications(api);
var params = {'buyer_team__reference':
67890};
applications.getSpecific('12345', params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'application': {
    'contractor_ciphertext':
'~012345abcdef',
    'contractor_name': 'John Johnson',
    'contractor_portrait_url':
'https://...',
    'contractor_ref': '675604',
    'contractor_title': 'Software
developer - test WEBAPI oAuth',
    'cover_letter': '',
    'created_type': 'Client',
    'ctime_epoch': '1404794618',
    'engagement_duration_ref': '2',
    'feedback_score': '0',
    'fp_amount_agreed': '',
    'fp_pay_amount': '0',
    'fp_upfront_payment': '0',
    'hr_charge_rate': '22.22',
    'hr_pay_rate': '20',
    'interview_status': 'Waiting For
Provider',
    'invite_letter': 'Test invite to
interview via oAuth',
    'is_auto_hidden': 'f',
    'is_hidden_by_buyer': '0',
```

```
        'is_hidden_by_provider': '0',
        'is_seen_by_buyer': 't',
        'is_shortlisted': '0',
        'is_undecided': '1',
        'job__description': 'Create job via
    oAuth api',
        'job_pref_matches': {'prefs_match':
    '0', 'prefs_total': '0'},
        'job_type': 'Hourly',
        'offer_status': '',
        'opening_ciphertext':
    '~00012345abcde',
        'rid': '12345',
        'status': 'In Process',
        'team_nid': 'unazcm8kj7ytdrwwgc-pkq',
        'total_hours': '0',
        'ui_opening_title': 'Test oAuth API
    create job'
      },
      'auth_user': {
        'first_name': 'John',
        'last_name': 'Johnson',
        'timezone': 'Asia/Omsk',
        'timezone_offset': '25200'
      },
      'server_time': '1405392072'
    }
```

List job applications as freelancer

Endpoint

GET /api/hr/v3/contractors/applications.{format}

This call lists all job applications made by a freelancer.

Required key permissions

View your job applications

DEFINITION

```
applications.getList(params, callback);
```

EXAMPLE REQUEST

```
var Applications = require('upwork-
api/lib/routers/hr/freelancers/applicatio
ns.js').Applications;
```

Arguments

format: optional, string

Default: `xml`

Valid values: `json, xml`

Response format.

Parameters

status: optional, string

Valid values: `interviews, invites, active`

The current status of the job application. If no value is specified, it will return applications with all statuses.

Error messages

401: Unauthorized

Possible output fields

(Click here to show/hide)

Returns

See “Possible output fields” section.

Get job application as freelancer

Endpoint

```
var applications = new Applications(api);
var params = {};
applications.getList(params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE


```
{
  'applications': [
    {
      'contractor_ciphertext':
'~012345abcdef',
      'ctime_epoch': '1404794618',
      'interview_status': 'Waiting For
Provider',
      'opening_ciphertext':
'~012345abcdef',
      'opening_title': 'Test oAuth API
create job',
      'rid': '12345',
      'status': 'In Process'
    }
  ],
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': '1405399432'
}
```


DEFINITION

GET

/api/hr/v3/contractors/applications/{application_id}.{format}

This call retrieves details about a specific job application made by a freelancer.

Required key permissions

View your job applications

Arguments

application_id:	**required, string** The job application ID.
format:	**optional, string** **Default:** `xml` **Valid values:** `json, xml` Response format.

Error messages

401:	Unauthorized
403:	No access to team/company/job/application

Possible output fields

(Click here to show/hide)

Returns

See “Possible output fields” section.

```
applications.getSpecific(reference,
callback);
```

EXAMPLE REQUEST

```
var Applications = require('upwork-
api/lib/routers/hr/freelancers/applicatio
ns.js').Applications;

var applications = new Applications(api);
applications.getSpecific('12345',
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'application': {
    'contractor_ciphertext':
'~012345abcdef',
    'contractor_name': 'John Johnson',
    'contractor_portrait_url':
'https://...',
    'contractor_ref': '675604',
    'contractor_title': 'Software
developer - test WEBAPI oAuth',
    'cover_letter': '',
    'created_type': 'Client',
    'ctime_epoch': '1404794618',
    'engagement_duration_ref': '2',
    'feedback_score': '0',
    'fp_amount_agreed': '',
    'fp_pay_amount': '0',
    'fp_upfront_payment': '0',
    'hr_charge_rate': '22.22',
    'hr_pay_rate': '20',
    'interview_status': 'Waiting For
Provider',
    'invite_letter': 'Test invite to
interview via oAuth',
```

```
        'is_auto_hidden': 'f',
        'is_hidden_by_buyer': '0',
        'is_hidden_by_provider': '0',
        'is_seen_by_buyer': 't',
        'is_shortlisted': '0',
        'is_undecided': '1',
        'job__description': 'Create job via
    oAuth api',
        'job_pref_matches': {'prefs_match':
    '0', 'prefs_total': '0'},
        'job_type': 'Hourly',
        'offer_status': '',
        'opening_ciphertext':
    '~012345abcdef',
        'rid': '12345',
        'status': 'In Process',
        'team_nid': 'unazcm8kj7ytdrwwgc-pkq',
        'total_hours': '0',
        'ui_opening_title': 'Test oAuth API
    create job'
      },
      'auth_user': {
        'first_name': 'John',
        'last_name': 'Johnson',
        'timezone': 'Asia/Omsk',
        'timezone_offset': '25200'
      },
      'server_time': '1405399734'
    }
```

List engagements

Endpoint

GET /api/hr/v2/engagements.{format}

This call returns engagement(s) based on the parameters supplied in the API call.

DEFINITION

```
engagements.getList(params, callback);
```

EXAMPLE REQUEST

```
var Engagements = require('upwork-
```

Required key permissions

View your contracts

Arguments

format:	**optional, string** **Default:** `xml` **Valid values:** `json, xml` Response format.

Parameters

buyer_team__ref erence:	**optional, string** The reference ID of the client's team. Example: `34567`. Use 'List teams' API call to get it.
include_sub_tea ms:	**optional, integer** **Valid values:** `0, 1` If set to `1`: the response includes info about sub teams.
provider__refere nce:	**optional, string** The freelancer's reference ID. Example: `1234`.
profile_key:	**optional, string** The unique profile key. It is used if the `provider_reference` param is absent. Example: `~~677961dcd7f65c01`.
job__reference:	**optional, string** The job reference ID. Use `List jobs` call to get it.

```
    api/lib/routers/hr/engagements.js').Engag
    ements;

    var engagements = new Engagements(api);
    var params = {};
    engagements.getList(params,
    function(error, data) {
      console.log(data);
    });
```

EXAMPLE RESPONSE


```
{
  'engagement': {
    'buyer_team__id':
'unazcm8kj7ytdrwwgc-pkq',
    'buyer_team__reference': '377329',
    'created_time': '1405434052000',
    'description': '',
    'dev_recno_ciphertext':
'~012345abcdef',
    'engagement_end_date': '',
    'engagement_job_type': 'hourly',
    'engagement_start_date':
'1405382400000',
    'engagement_title': 'Test offers API
send offer',
    'hourly_charge_rate': '20.0',
    'is_paused': '',
    'job__title': 'Test offers API send
offer',
    'job_ref_ciphertext':
'~012345abcdef',
    'offer__reference': '',
    'portrait_url': 'https://...',
    'provider__id': 'john_freelancer',
    'provider__reference': '675604',
    'provider_team__id': '',
    'provider_team__reference': '',
    'reference': '167747',
    'status': 'active',
    'weekly_hours_limit': '',
    'weekly_salary_pay_amount': '',
```

agency_team__reference:	**optional, string** The reference ID of the agency.
status:	**optional, string** **Valid values:** `active, closed` The current status of the engagement. Multiple statuses can be listed using semicolon. Example: `status=active;closed`.
created_time_from:	**optional, string** Filters by 'from' time. Example: `created_time_from=2008-09-09T00:00:01`.
created_time_to:	**optional, string** Filters by 'to' time. Example: `created_time_to=2009-01-20 11:59:59`.
page:	**optional, string** Pagination, formed as `$offset;$count`. Example: `page=20;10`
order_by:	**optional, string** Sorts results in format `$field_name1;$field_name2;..$field_nameN;AD...A`. Here `A` stands for ascending order, `D` - descending order. Valid field names for ordering are: `reference`, `created_time`, `offer__reference`, `job__reference`,


```
      'weekly_stipend_hours': ''
    },
    'lister': {
      'paging': {'count': '20', 'offset':
'0'},
      'query': '',
      'sort': {'sort': {'sort':
['created_time', 'asc']}},
      'total_count': '1',
      'total_items': '2'
    }
  }
```

`client_team__reference`,
`provider__reference`, `status`,
`engagement_start_date`,
`engagement_end_date`.

Error messages

401: Unauthorized

403: Forbidden

Possible output fields

(Click here to show/hide)

Returns

See “Possible output fields” section.

Get engagement

Endpoint

GET

/api/hr/v2/engagements/{engagement_reference}.{format}

This call retrieves details about a specific engagement.

Required key permissions

View your contracts

Arguments

engagement_refe rence:	**required, integer** The reference ID of the engagement.
format:	**optional, string**

DEFINITION

```
engagements.getSpecific(reference,
callback);
```

EXAMPLE REQUEST

```
var Engagements = require('upwork-
api/lib/routers/hr/engagements.js').Engag
ements;

var engagements = new Engagements(api);
engagements.getSpecific('12345',
function(error, data) {
  console.log(data);
});
```

Default: `xml`

Valid values: `json, xml`

Response format.

Error messages

401:	Unauthorized
403:	Forbidden
400:	Wrongly requested data

Possible output fields

(Click here to show/hide)

Returns

Returns information about the specified engagement.

EXAMPLE RESPONSE


```
{
  'buyer_team__id': 'unazcm8kj7ytdrwwgc-
pkq',
  'buyer_team__reference': '377329',
  'created_time': '1405434052000',
  'description': '',
  'dev_recno_ciphertext':
'~012345abcdef',
  'engagement_end_date': '',
  'engagement_job_type': 'hourly',
  'engagement_start_date':
'1405382400000',
  'engagement_title': 'Test offers API
send offer',
  'hourly_charge_rate': '20.0',
  'is_paused': '',
  'job__title': 'Test offers API send
offer',
  'job_ref_ciphertext':
'~01234567aaabbb',
  'offer__reference': '',
  'portrait_url': 'https://...',
  'provider__id': 'john_freelancer',
  'provider__reference': '675604',
  'provider_team__id': '',
  'reference': '167747',
  'status': 'active',
  'weekly_hours_limit': '',
  'weekly_salary_pay_amount': '',
  'weekly_stipend_hours': '',
  'milestones': ''
}
```


Suspend contract

Endpoint

PUT

DEFINITION

```
contracts.suspendContract(reference,
```

/api/hr/v2/contracts/{contract_reference}/suspend.{format}

This call allows you to suspend a contract.

Required key permissions

Close your contracts

Arguments

contract_referenc e:	**required, string** The reference ID of the contract. It's the engagement reference ID.
format:	**optional, string** **Default:** `xml` **Valid values:** `json, xml` Response format.

Parameters

message:	**optional, string** A message for the contractor.

Error messages

401:	Unauthorized
403:	Forbidden

Returns

If successful, this resource returns a `200 OK` message.

Restart contract

Endpoint

```
params, callback);
```

EXAMPLE REQUEST

```
var Contracts = require('upwork-
api/lib/routers/hr/contracts.js').Contrac
ts;

var contracts = new Contracts(api);
var reference = '12345';
var params = {'message': 'Suspended the
contract'};
contracts.suspendContract(reference,
params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'message': 'OK',
  'server_time': 1405516669
}
```

DEFINITION

PUT

/api/hr/v2/contracts/{contract_reference}/restart.{format}

This call allows you to restart a contract.

Required key permissions

Close your contracts

Arguments

contract_referenc e:	**required, string** The reference ID of the contract. It's the engagement reference ID.
format:	**optional, string** **Default:** `xml` **Valid values:** `json, xml` Response format.

Parameters

message:	**optional, string** A message for the contractor.

Error messages

401:	Unauthorized
403:	Forbidden

Returns

If successful, this resource returns a `200 OK` message.

End contract

```
contracts.restartContract(reference,
params, callback);
```

EXAMPLE REQUEST

```
var Contracts = require('upwork-
api/lib/routers/hr/contracts.js').Contrac
ts;

var contracts = new Contracts(api);
var reference = '12345';
var params = {'message': 'Restarting the
contract'};
contracts.restartContract(reference,
params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'message': 'OK',
  'server_time': 1405516669
}
```

DEFINITION

Warning

Currently, Upwork website does not show scores sent in the `fb_scores` field. We strongly suggest you to use Upwork user interface to provide scores.

Endpoint

DELETE /api/hr/v2/contracts/{contract_reference}.{format}

This call allows you to close a contract.

Required key permissions

Close your contracts

Arguments

contract_reference:	**required, string** The reference ID of the contract. It's the engagement reference ID.
format:	**optional, string** **Default:** `xml` **Valid values:** `json, xml` Response format.

Parameters

reason:	**required, string** **Valid values:** `API_REAS_MISREPRESENTED_SKILLS, API_REAS_CONTRACTOR_NOT_RESPONSIVE, API_REAS_HIRED_DIFFERENT, API_REAS_JOB_COMPLETED_SUCCESSF`

DEFINITION

```
contracts.endContract(reference, params,
callback);
```

EXAMPLE REQUEST

```
var Contracts = require('upwork-
api/lib/routers/hr/contracts.js').Contrac
ts;

var contracts = new Contracts(api);
var reference = '12345';
var params = {'reason':
'API_REAS_WORK_NOT_NEEDED'};
contracts.endContract(reference, params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'message': 'OK',
  'server_time': 1405516669
}
```

`ULLY, API_REAS_WORK_NOT_NEEDED, API_REAS_UNPROFESSIONAL_CONDUCT`

The reason key. Reasons descriptions: `API_REAS_MISREPRESENTED_SKILLS` - 'Freelancer misrepresented his/her skills', `API_REAS_CONTRACTOR_NOT_RESPONSIVE` - 'Freelancer not responsive', `API_REAS_HIRED_DIFFERENT` - 'Hired a different freelancer', `API_REAS_JOB_COMPLETED_SUCCESSFULLY` - 'Job was completed successfully', `API_REAS_WORK_NOT_NEEDED` - 'No longer need this work completed', `API_REAS_UNPROFESSIONAL_CONDUCT` - 'Unprofessional conduct'.

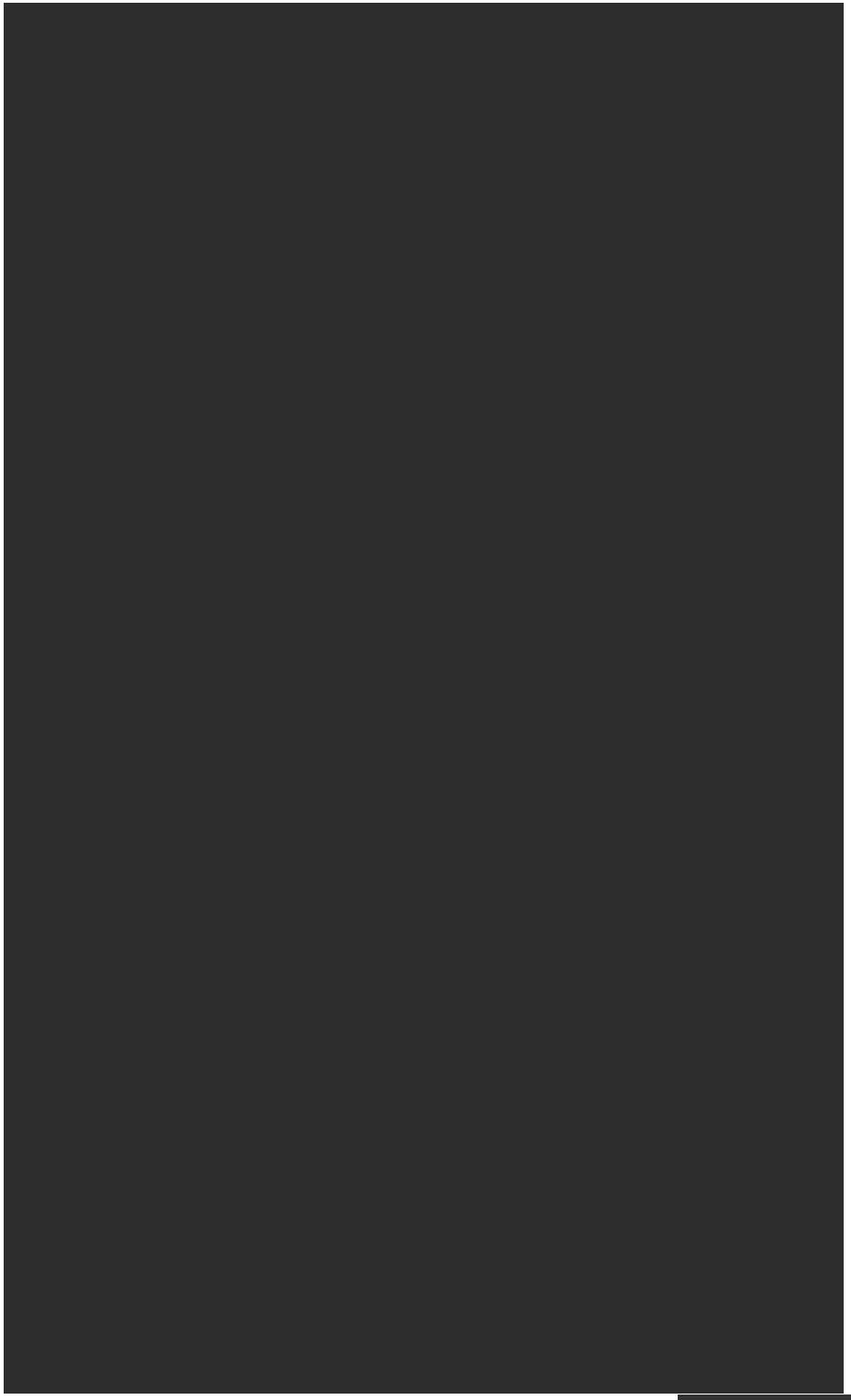
would_hire_agai n: **required, string**

Valid values: `yes, no`

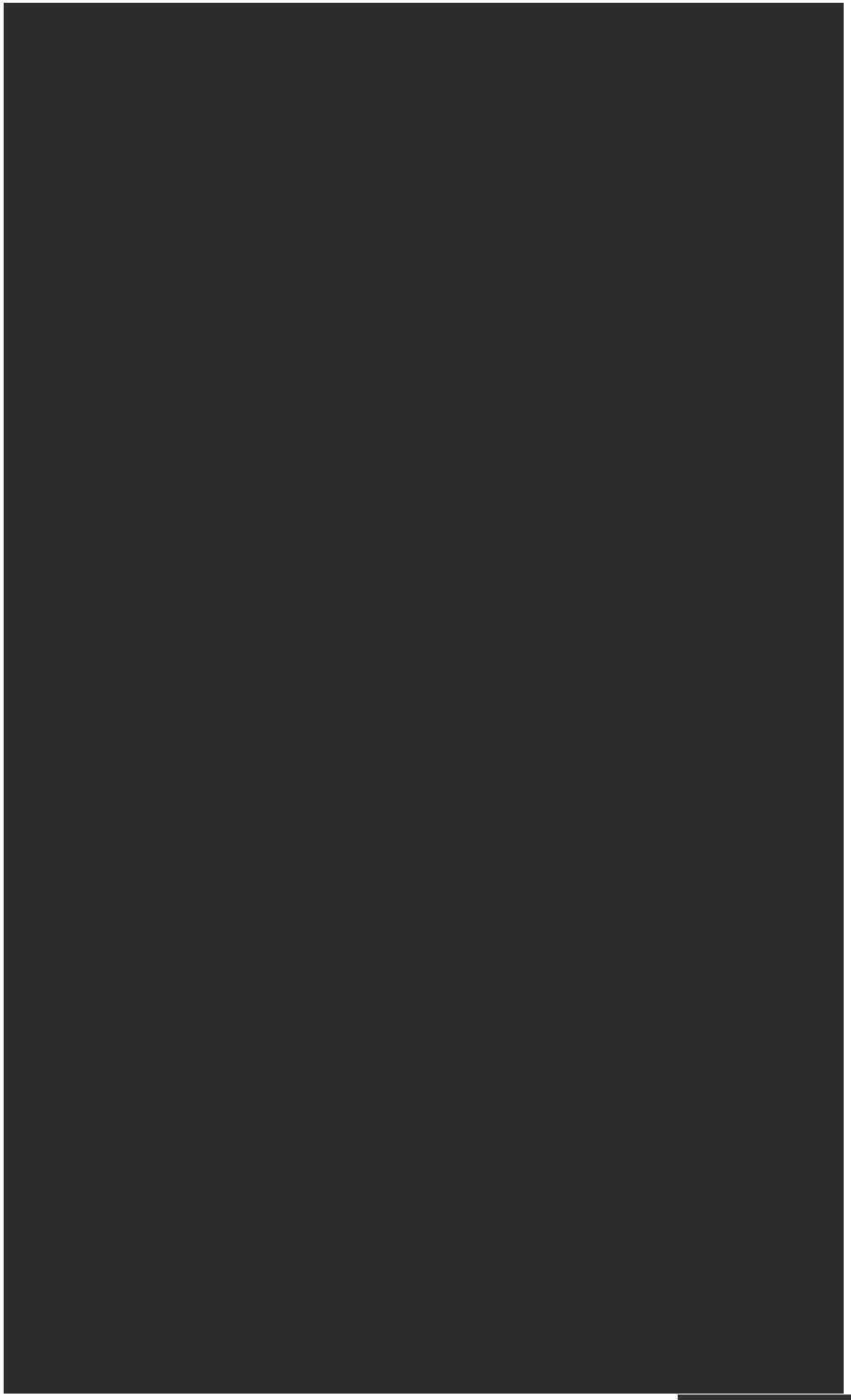
Indicates whether you would hire a freelancer again. Required if the total charge on the contract is $0 (zero dollars). Valid values are: `yes`, `no`.

fb_scores: **conditionally required, string**

Estimate, it could be an array of



scores, where ID is reference to the score description. The feedback scores are optional, but if present they must be complete: all scores. Valid array keys for freelancer are: `3` - 'Skills / competency and skills for the job, understanding of specifications/instructions'; `4` - 'Quality / quality of work deliverables'; `5` - 'Availability / online presence on a consistent schedule'; `6` - 'Deadlines / ability to complete tasks on time'; `7` - 'Communication / communication skills, frequent progress updates, responsiveness'; `8` - 'Cooperation / cooperation and flexibility, suggestions for improvement'. Valid array keys for employer are: `9` - 'Skills / competency and skills for the job, understanding of task complexities'; `10` - 'Quality / quality of specifications/instructions'; `11` - 'Availability / online presence on a consistent schedule'; `12` - 'Deadlines / understanding of complexities and trade-offs'; `13` - 'Communication / communication



skills and responsiveness, feedback and guidance'; `14` - 'Cooperation / cooperation and flexibility, open to suggestions for improvement'. In the following example 1,2,3,7,8,11 are references, while 5 is the score: `reason=API_REAS_WORK_NOT_NEEDED&would_hire_again=yes&fb_comment=test%20comment&fb_scores[1]=5&fb_scores[2]=5&fb_scores[3]=5&fb_scores[7]=5&fb_scores[8]=5&fb_scores[11]=5`.

fb_comment: **conditionally required, string**

A feedback comment. This comment is optional, but if present, then the scores are required. Example: `This is my nice feedback`.

fb_private: **required, integer**

A private feedback score (from 1 to 11) for freelancer.

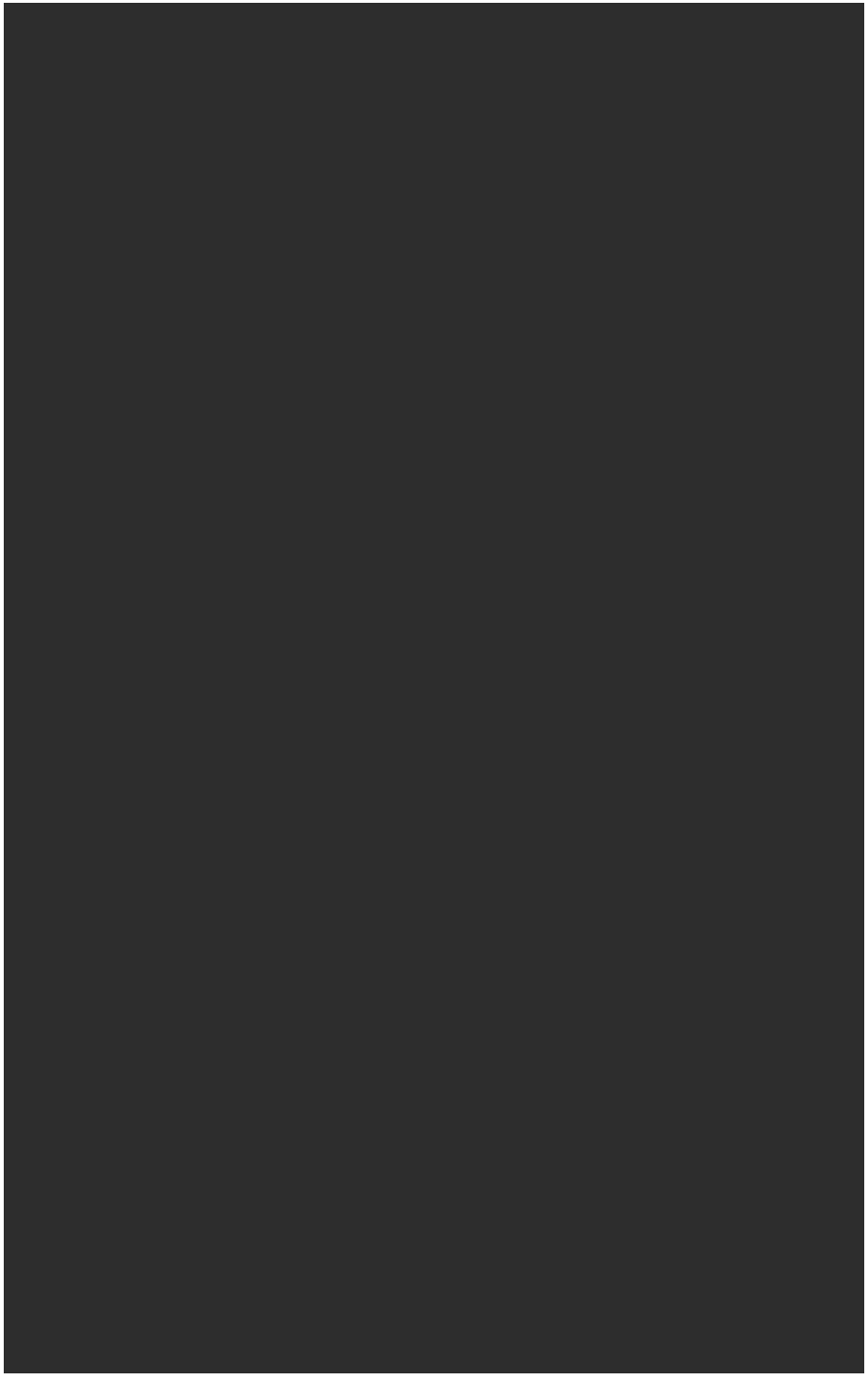
Error messages

401: Unauthorized

403: Forbidden

Returns

If successful, this resource returns a `200 OK` message.



Milestones

Fixed-priced jobs allow splitting the contract into milestones (significant points in the process of the work to be done). Then, clients are charged when a milestone is created, the money is placed into “escrow”, and released when the milestone is completed.

As a client, once you’ve hired a freelancer on a fixed-price contract, you can start adding milestones. Milestones are created in a “not funded” status. You can activate one milestone at a time. Also note, that only milestones with a “not funded” status can be edited or deleted. Once the milestone is active it can only be closed by approving it or ending the contract, which cancels it.

As a freelancer you can submit work to a milestone. This work submission can be either approved or rejected by a client. If the submission is approved, the milestone proceeds to a “paid” status and funds in escrow are released to the freelancer.

Get active milestone

Endpoint

GET

/api/hr/v3/fp/milestones/statuses/active/contracts/{contract_reference}.{format}

DEFINITION

```
milestones.getActiveMilestone(contractId,
callback);
```

This API call allows a Hiring Manager to get an active milestone of specific contract. Note that the user must be authorized in Upwork and must be a Hiring Manager in the team to be able to access the data.

Required key permissions

View milestones and submissions on your behalf

Arguments

format:	**optional, string** **Default:** `json` **Valid values:** `json` Response format.
contract_reference:	**required, integer** Contract reference. Contracts info are available in the Engagements API.

Error messages

401:	Unauthorized
403:	Forbidden
403:	No recruiter or hiring manager permissions
403:	No access to team/company

Possible output fields

(Click here to show/hide)

Returns

Returns an object representing the currently active milestone.

EXAMPLE REQUEST

```
var Milestones = require('upwork-
api/lib/routers/hr/milestones.js').Milest
ones;

var milestones = new Milestones(api);
var contractId = 1234;
milestones.getActiveMilestone(contractId,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405516669,
  'milestone': {
    'contract_id': '214794',
    'deposit_amount': '30',
    'description': 'New description of
milestone 2',
    'due_date': '2015-01-01 00:00:00',
    'id': '11206'
  }
}
```

List submissions for a milestone

Endpoint

GET

/api/hr/v3/fp/milestones/{milestone_id}/submissions.{format}

This API call allows a Hiring Manager to get all submissions for specific milestone. Note that the user must be authorized in Upwork and must be a Hiring Manager in the team to be able to access the data.

Required key permissions

View milestones and submissions on your behalf

Arguments

format:	**optional, string** **Default:** `json` **Valid values:** `json` Response format.
milestone_id:	**required, integer** Milestone reference.

Error messages

401:	Unauthorized
403:	Forbidden
403:	No recruiter or hiring manager permissions
403:	No access to team/company

DEFINITION

```
milestones.getSubmissions(milestoneId,
callback);
```

EXAMPLE REQUEST

```
var Milestones = require('upwork-
api/lib/routers/hr/milestones.js').Milest
ones;

var milestones = new Milestones(api);
var milestoneId = 1234;
milestones.getSubmissions(milestoneId,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405516669,
  'submissions': [{u'id': u'3378'},
{u'id': u'3380'}]
}
```

Possible output fields

(Click here to show/hide)

Returns

Returns a list of submissions' IDs.

Create a milestone

Endpoint

POST /api/hr/v3/fp/milestones.{format}

This API call allows a Hiring Manager to create a milestone. Note that the user must be authorized in Upwork and must be Hiring Manager in the team to be able to create a milestone.

Required key permissions

Modify milestones and submissions on your behalf

Arguments

format:	**optional, string** **Default:** `json` **Valid values:** `json` Response format.

Parameters

contract_reference:	**required, integer** Contract reference. Contracts info are available in the Engagements API.

DEFINITION

```
milestones.create(params, callback);
```

EXAMPLE REQUEST

```
var Milestones = require('upwork-api/lib/routers/hr/milestones.js').Milestones;

var milestones = new Milestones(api);
var params = {
  'contract_reference': '1234',
  'milestone_description': 'First milestone',
  'deposit_amount': '50',
  'due_date': '12-12-2014'
};
milestones.create(params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
```

milestone_descri ption:	**required, string** Name of the milestone.
deposit_amount:	**required, string** Amount to deposit for this milestone.
due_date:	**optional, string** Expected date of finalization. Format mm-dd-yyyy.

Error messages

401:	Unauthorized
403:	Forbidden
403:	No recruiter or hiring manager permissions
403:	No access to team/company
400:	Contract id must be given
400:	The function cannot be used to create the first milestone
400:	Date is earlier or equal than milestone {id}

Returns

This resource returns the ID of the milestone created.

Edit a milestone

Endpoint

PUT /api/hr/v3/fp/milestones/{milestone_id}.{format}

This API call allows a Hiring Manager to edit a milestone. Note that the user must be authorized in Upwork and must

```
      'last_name': 'Johnson',
      'timezone': 'Asia/Omsk',
      'timezone_offset': '25200'
    },
    'id': '9974',
    'server_time': 1405516669
  }
```

DEFINITION

```
milestones.edit(milestoneId, params,
callback);
```

EXAMPLE REQUEST

be a Hiring Manager in the team to be able to edit the milestone.

Required key permissions

Modify milestones and submissions on your behalf

Arguments

format:	**optional, string** **Default:** `json` **Valid values:** `json` Response format.
milestone_id:	**required, integer** Milestone reference.

Parameters

milestone_descri ption:	**optional, string** Name of the milestone.
deposit_amount:	**optional, string** Amount to deposit for this milestone.
message:	**optional, string** Message from the client to the freelancer.
due_date:	**optional, string** Expected date of finalization. Format mm-dd-yyyy.

Error messages

401:	Unauthorized
403:	Forbidden
403:	No recruiter or hiring manager

```
var Milestones = require('upwork-
api/lib/routers/hr/milestones.js').Milest
ones;

var milestones = new Milestones(api);
var milestoneId = 1234;
var params = {
  'milestone_description': 'New
description',
  'deposit_amount': '30',
  'due_date': '01-01-2015',
  'message': 'Updating the milestone
terms'
};
milestones.edit(milestoneId, params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'message': 'ok',
  'server_time': 1405516669
}
```

permissions

403: No access to team/company

400: This function cannot be used to first the first milestone

400: Only not_funded milestones can be edited

Returns

Returns 200 OK in case of success.

Activate a milestone

Endpoint

PUT /api/hr/v3/fp/milestones/{milestone_id}/activate.{format}

This API call allows a Hiring Manager to activate a Milestone. Note that the user must be authorized in Upwork and must be a Hiring Manager in the team to be able to activate the milestone.

Required key permissions

Modify milestones and submissions on your behalf

Arguments

format: **optional, string**

Default: `json`

Valid values: `json`

Response format.

DEFINITION

```
milestones.activate(milestoneId, params,
callback);
```

EXAMPLE REQUEST

```
var Milestones = require('upwork-
api/lib/routers/hr/milestones.js').Milest
ones;

var milestones = new Milestones(api);
var milestoneId = 1234;
var params = {
  'message': 'Activating second
milestone'
};
milestones.activate(milestoneId, params,
function(error, data) {
  console.log(data);
});
```

milestone_id: **required, integer**

Milestone reference.

Parameters

message: **optional, string**

Message from the client to the freelancer.

Error messages

401:	Unauthorized
403:	Forbidden
403:	No recruiter or hiring manager permissions
403:	No access to team/company
400:	This function cannot be used to activate the first milestone
400:	Invalid state '{state}'
400:	Invalid previous milestone state '{state}'

Returns

Returns 200 OK in case of success.

Approve a milestone

Endpoint

PUT /api/hr/v3/fp/milestones/{milestone_id}/approve.{format}

Closes the milestone by paying the deposit amount if no amount is specified.

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'message': 'ok',
  'server_time': 1405516669
}
```

DEFINITION

```
milestones.approve(milestoneId, params,
callback);
```

EXAMPLE REQUEST

Required key permissions

Modify milestones and submissions on your behalf

Arguments

format: **optional, string**

Default: `json`

Valid values: `json`

Response format.

milestone_id: **required, integer**

Milestone reference.

Parameters

amount: **required, integer**

Amount of money to be paid. If none provided, the full deposit_amount is paid.

bonus: **optional, integer**

Amount of money paid as bonus.

pay_comments: **optional, string**

Comments on the payment.

underpayment_reason: **optional, string**

Valid values: `329, 330, 331, 332`

Reason for a smaller payment than the one agreed. Values description: `329` - Payment lower due to freelancer performance; `330` - Freelancer is performing well, but the scope of work changed; `331` - Freelancer is performing well, but I

```
var Milestones = require('upwork-
api/lib/routers/hr/milestones.js').Milest
ones;

var milestones = new Milestones(api);
var milestoneId = 1234;
var params = {
  'amount', '10',
  'bonus', '5',
  'pay_comments', 'A comment',
  'underpayment_reason', '331',
  'note2contractor', 'A note here'
};
milestones.approve(milestoneId, params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'message': 'ok',
  'server_time': 1405516669
}
```

decided to make an earlier payment; `332` - Payment is lower for other reasons, but freelancer is performing well.

note2contractor: **optional, string**
Notes from the client to the freelancer.

Error messages

401:	Unauthorized
403:	Forbidden
403:	No recruiter or hiring manager permissions
403:	No access to team/company
400:	Invalid state when approving the milestone: '{state}'

Returns

Returns 200 OK in case of success.

Delete a milestone

Endpoint

DELETE /api/hr/v3/fp/milestones/{milestone_id}.{format}

This API call allows a Hiring Manager to delete a milestone. Note that the user must be authorized in Upwork and must be a Hiring Manager in the team to be able to delete the milestone. Only milestondes with status `not_funded` can

DEFINITION

```
milestones.delete(milestoneId, callback);
```

EXAMPLE REQUEST

```
var Milestones = require('upwork-
api/lib/routers/hr/milestones.js').Milest
ones;
```

be deleted.

Required key permissions

Modify milestones and submissions on your behalf

Arguments

format:	**optional, string** **Default:** `json` **Valid values:** `json` Response format.
milestone_id:	**required, integer** Milestone reference.

Error messages

401:	Unauthorized
403:	Forbidden
403:	No recruiter or hiring manager permissions
403:	No access to team/company
400:	Invalid state '{state}'

Returns

Returns 200 OK in case of success.

Submit work to a milestone

Endpoint

POST /api/hr/v3/fp/submissions.{format}

This API call allows a freelancer to submit work to a

```
var milestones = new Milestones(api);
milestoneId = 1234;
milestones.delete(milestoneId,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'message': 'ok',
  'server_time': 1405516669
}
```

DEFINITION

```
submissions.requestApproval(params,
callback);
```

milestone to be approved by the client. Note that the user must be authorized in Upwork and must be a freelancer of a contract or AM of such freelancer.

Required key permissions

Modify milestones and submissions on your behalf

Arguments

format: **optional, string**

Default: `json`

Valid values: `json`

Response format.

Parameters

milestone_id: **required, integer**

Milestone reference.

note2client: **required, string**

Notes from freelancer to client about work that was done.

amount: **required, integer**

Amount requested by the freelancer.

Error messages

401: Unauthorized

403: Forbidden

403: No access to submit approval

Returns

Returns 200 OK in case of success.

EXAMPLE REQUEST

```
var Submissions = require('upwork-
api/lib/routers/hr/submissions.js').Submi
ssions;

var submissions = new Submissions(api);
var params = {
  'milestone_id': '1234',
  'note2client': 'First milestone
completed',
  'amount': '15'
};
submissions.requestApproval(params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'message': 'ok',
  'server_time': 1405516669
}
```

Approve a milestone submission

Endpoint

PUT

/api/hr/v3/fp/submissions/{submission_id}/approve.{format}

This API call allows a Hiring Manager to approve the submission work to a milestone made by the freelancer. Note that the user must be authorized in Upwork and must be a Hiring Manager in the team to be able to approve a milestone submission.

Required key permissions

Modify milestones and submissions on your behalf

Arguments

format: optional, string

Default: `json`

Valid values: `json`

Response format.

submission_id: required, integer

Submission reference.

Parameters

amount: required, integer

Amount of money to be paid.

bonus: optional, integer

Amount of money paid as bonus.

pay_comments: optional, string

DEFINITION

```
submissions.approve(submissionId, params,
callback);
```

EXAMPLE REQUEST

```
var Submissions = require('upwork-
api/lib/routers/hr/submissions.js').Submi
ssions;

var submissions = new Submissions(api);
var submissionId = 1234;
var params = {
  'amount' => '10',
  'bonus' => '3',
  'pay_comments' => 'A comment',
  'underpayment_reason' => '331',
  'note2contractor' => 'Note to
freelancer'
};
submissions.approve(submissionId, params,
data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'message': 'ok',
  'server_time': 1405516669
}
```

Comments on payment.

underpayment_reason: **optional, string**

Valid values: `329, 330, 331, 332`

Reason for a smaller payment than the one agreed. Values description: `329` - Payment lower due to freelancer performance; `330` - Freelancer is performing well, but the scope of work changed; `331` - Freelancer is performing well, but I decided to make an earlier payment; `332` - Payment is lower for other reasons, but freelancer is performing well.

note2contractor: **optional, string**

Notes from the client to the freelancer.

Error messages

401:	Unauthorized
403:	Forbidden
403:	No recruiter or hiring manager permissions
403:	No access to team/company
400:	Invalid state when approving the milestone: '{state}'

Returns

Returns 200 OK in case of success.

Reject a milestone submission

Endpoint

PUT

/api/hr/v3/fp/submissions/{submission_id}/reject.{format}

This API call allows a Hiring Manager to reject the submission of work to a milestone. Note that the user must be authorized in Upwork and must be a Hiring Manager in the team to be able to reject the submission.

Required key permissions

Modify milestones and submissions on your behalf

Arguments

format: **optional, string**

Default: `json`

Valid values: `json`

Response format.

submission_id: **required, integer**

Submission reference.

Parameters

note2contractor: **required, string**

Notes from client to freelancer.

Error messages

401: Unauthorized

403: Forbidden

DEFINITION

```
submissions.reject(submissionId, params,
callback);
```

EXAMPLE REQUEST

```
var Submissions = require('upwork-
api/lib/routers/hr/submissions.js').Submi
ssions;

var submissions = new Submissions(api);
var submissionId = 1234;
var params = {
  'note2contractor': 'Note to freelancer'
};
submissions.reject(submissionId, params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'message': 'ok',
  'server_time': 1405516669
}
```

403: No recruiter or hiring manager permissions

403: No access to team/company

Returns

Returns 200 OK in case of success.

Payments

This section describes resources that allow you to make custom payments (bonuses) and list payments made.

Make custom payment

Endpoint

POST

/api/hr/v2/teams/{team_reference}/adjustments.{format}

This call is designed for making custom payments for a specific team and engagement in that team.

Required key permissions

Make one-time payments to your contractors

Arguments

team_reference: **required, integer**
The reference ID of the team.

format: **optional, string**

DEFINITION

```
payments.submitBonus(
  teamReference, params, callback);
```

EXAMPLE REQUEST

```
var Payments = require('upwork-
api/lib/routers/payments.js').Payments;

var payments = new Payments(api);
teamReference = '12345';
params = {
  'engagement__reference': 78910,
  'comment': 'Test bonus payment',
  'amount': 20
}
payments.submitBonus(teamReference,
params, function(error, data) {
  console.log(data);
```

Default: `xml`

Valid values: `json, xml`

Response format.

Parameters

engagement__reference:	**required, string** The engagement ID number of the Custom payment. You can obtain this ID number using the `List engagements` API call. Example: `34567`.
comments:	**required, string** A comment about the custom payment. Example: `Bonus for good job`.
charge_amount:	**required, number** The amount that is charged to the client. Example: `110`.
notes:	**optional, string** Private notes about the Custom Payment.

Error messages

401:	Unauthorized
403:	Forbidden

Returns

This resource returns the adjustment reference.

```
});
```

EXAMPLE RESPONSE

```
{'reference': '011223344'}
```

MC to Messages Transition

This section describes API transition from MC to Messages.

Below you can find the details and requirements for each API.

Messages

This section describes resources that allow you to get/update rooms and send messages.

Retrieve rooms information

Endpoint

GET /api/messages/v3/{company_user_id}/rooms.{format}

Gets the rooms for the current user and organization.

Required key permissions

Gets the rooms for the current user and organization

Arguments

company_user_id **required, string**

: The company ID or team ID, or user ID. Use 'List companies' or 'List teams' API calls to get the team/company IDs.

format: **optional, string**

Default: `json`

Response format.

Parameters

sortOrder: **optional, boolean**

Specifies the sort order of the

DEFINITION

EXAMPLE RESPONSE

```
{
  'auth_user': {
  # ..
  },
  'server_time': 123456790,
  'rooms': [
    {
      'roomId': 'room_12345',
      # ..
    }
    # ..
  ]
}
```

returned rooms. As of 7/29/2014, the valid values are `alphabetical` and `recentTimestamp` (the newest first). Default: alphabetical (by room name).

type: **optional, string**

Default: `all`

Valid values: `all, GROUP, ONE_ON_ONE, INTERVIEW`

Specifies a filter by type for the list of rooms.

cursor: **optional, string**

Valid values: `0, 1`

The offset into the list from which we want to start returning results. Used only for paginating through the result set. In this case, all returned room names must be lexicographically greater than the page token.

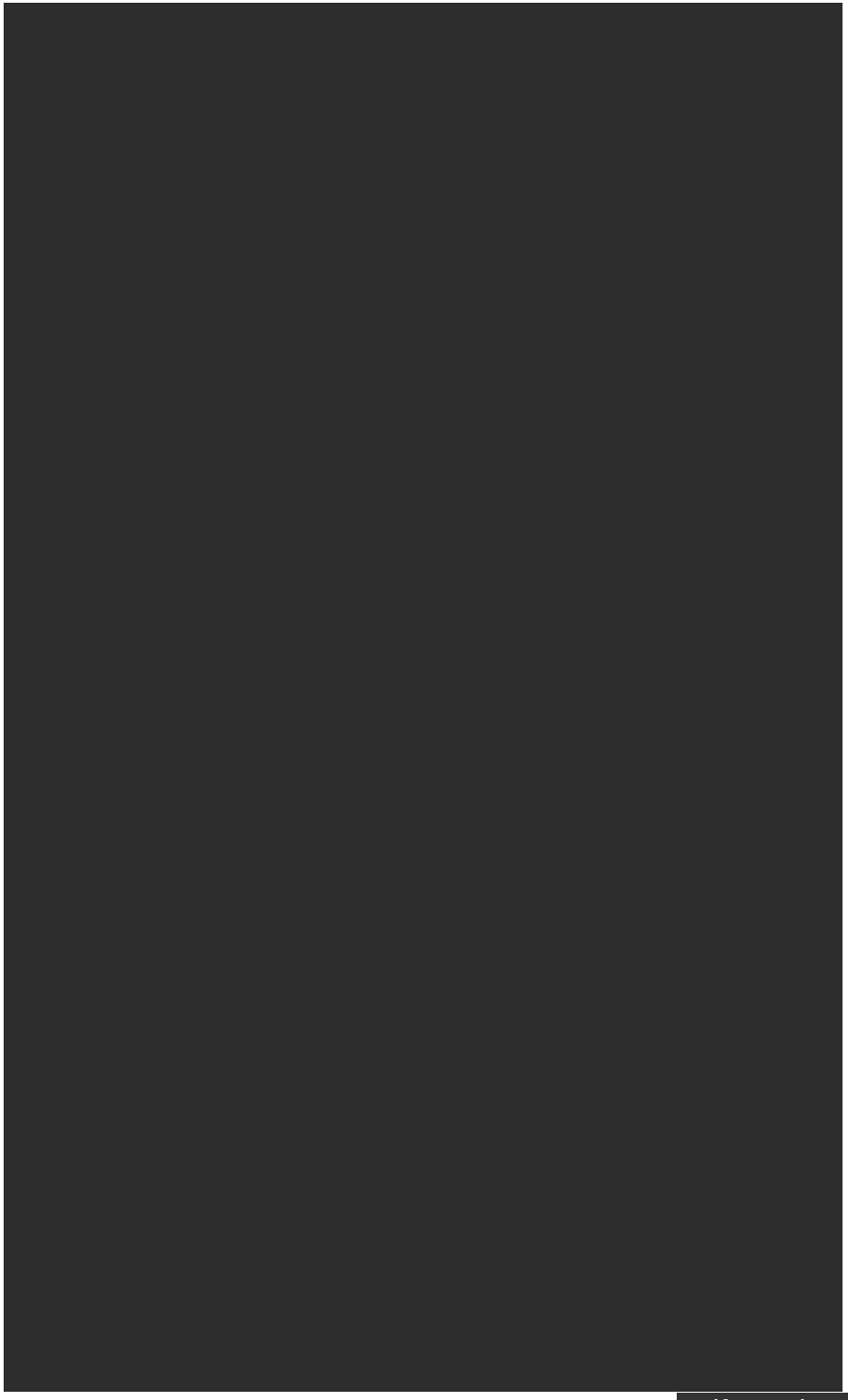
limit: **optional, integer**

Default: `100`

The maximum number of results to be returned. Used only for paginating through the result set. Maximum value: allowed: `1000`.

activeSince: **optional, string**

Indicates to return only the rooms that have had new activity since the given time, as well as favorites (which



are treated as always active).

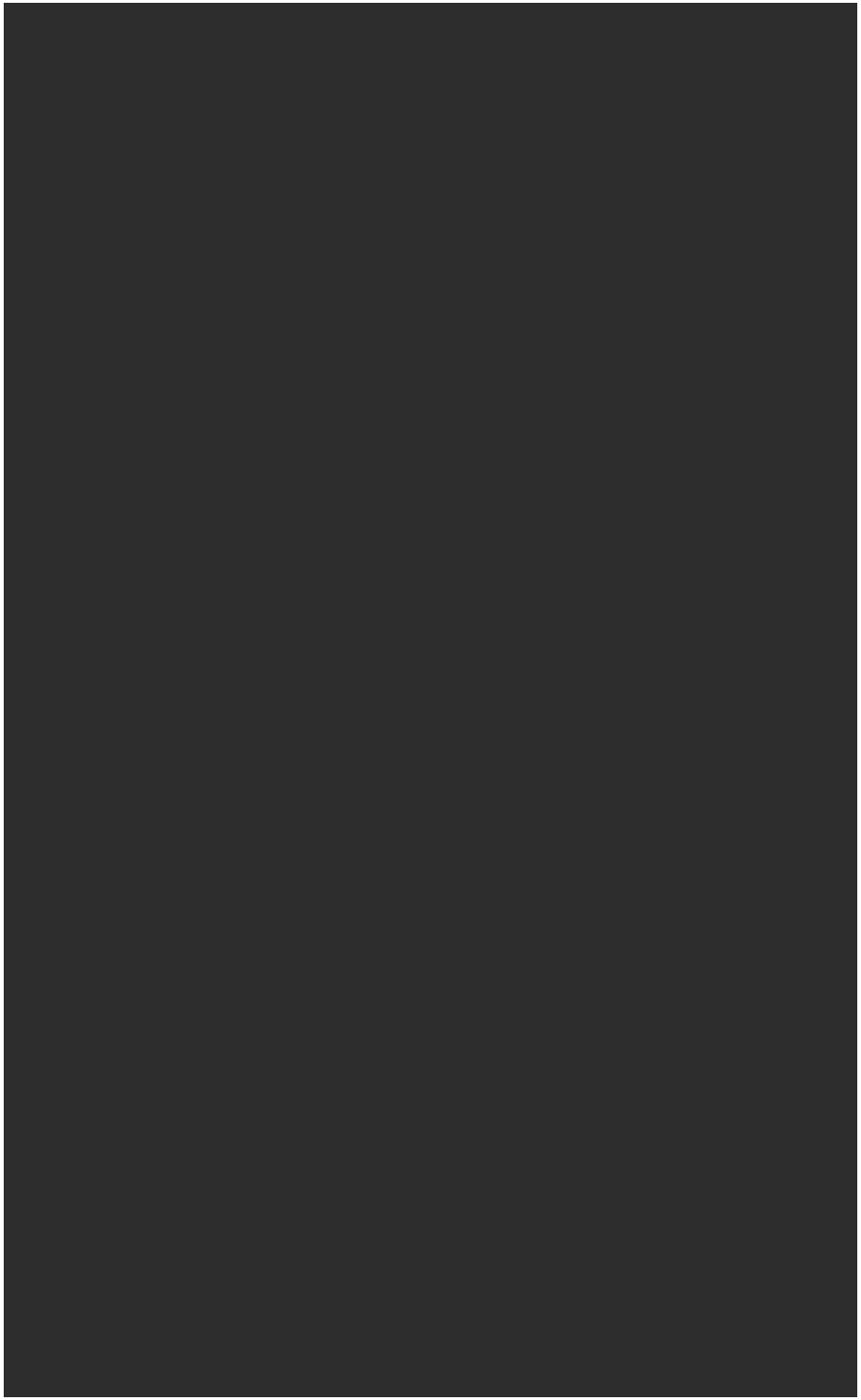
includeFavoritesIfActiveSinceSet: **optional, boolean**
Default: `true`
Valid values: `true, false`
Indicates to include all favorite rooms, regardless of when the last activity in those rooms took place. Ignored if `activeSince` parameter is not set.

includeUnreadIfActiveSinceSet: **optional, boolean**
Default: `true`
Valid values: `true, false`
Indicates to include all rooms that have unread stories, regardless of when the last activity in those rooms took place. Ignored if `activeSince` parameter is not set.

locale: **optional, string**
Default: `en_US`
When getting rooms for a user, the latest story is returned. This is the locale used for the latest story.

returnUsers: **optional, boolean**
Valid values: `true, false`
Indicates to return the users for the room too. Default: don't return the users.

includeHidden: **optional, boolean**



Default: `true`

Valid values: `true, false`

When set to false, it returns rooms which are not hidden, otherwise it returns all the rooms. Valid values are: true, false.

returnUserRoles: **optional, boolean**

Default: `true`

Valid values: `true, false`

Indicates whether or not the information about user roles should be fetched from the permissions service and returned in the response.

Error messages

400: Bad Request

401: Unauthorized

403: Forbidden

Possible output fields

(Click here to show/hide)

Returns

Returns the information on the rooms. Details on returned data:

Get a specific room information

Endpoint

GET

DEFINITION

EXAMPLE RESPONSE

/api/messages/v3/{company_user_id}/rooms/{room_id}.{format}

Get a specific room information.

Required key permissions

Gets the rooms for the current user and organization

Arguments

company_user_id	**required, string**
:	The company ID or team ID, or user ID. Use 'List companies' or 'List teams' API calls to get the team/company IDs.
format:	**optional, string**
	Default: `json`
	Response format.

Parameters

returnUserTimestamps:	**optional, boolean**
	Valid values: `true, false`
	Indicates whether or not the current user's timestamps and the first several stories in the room should be returned in the response.
returnStories:	**optional, boolean**
	Valid values: `true, false`
	Indicates whether the latest stories in the room should be returned.

```
{
  'auth_user': {
  # ..
  },
  'server_time': 123456790,
  'room': {
    'roomId': 'room_12345',
    'roomName': 'Room name',
    # ..
  }
}
```

returnUsers: **optional, boolean**

Valid values: `true, false`

Indicates whether the list of users in the room should be returned.

limit: **optional, integer**

Default: `100`

The maximum number of stories to be returned. Used only for paginating through the result set. Maximum value allowed: `1000`. Used only for paginating through the result set. This parameter is ignored if `returnTimestampsAndStories` is set to false

returnUserRoles: **optional, boolean**

Default: `true`

Valid values: `true, false`

Indicates whether the information about user roles should be fetched from the permissions service and returned in the response.

Error messages

401: Unauthorized

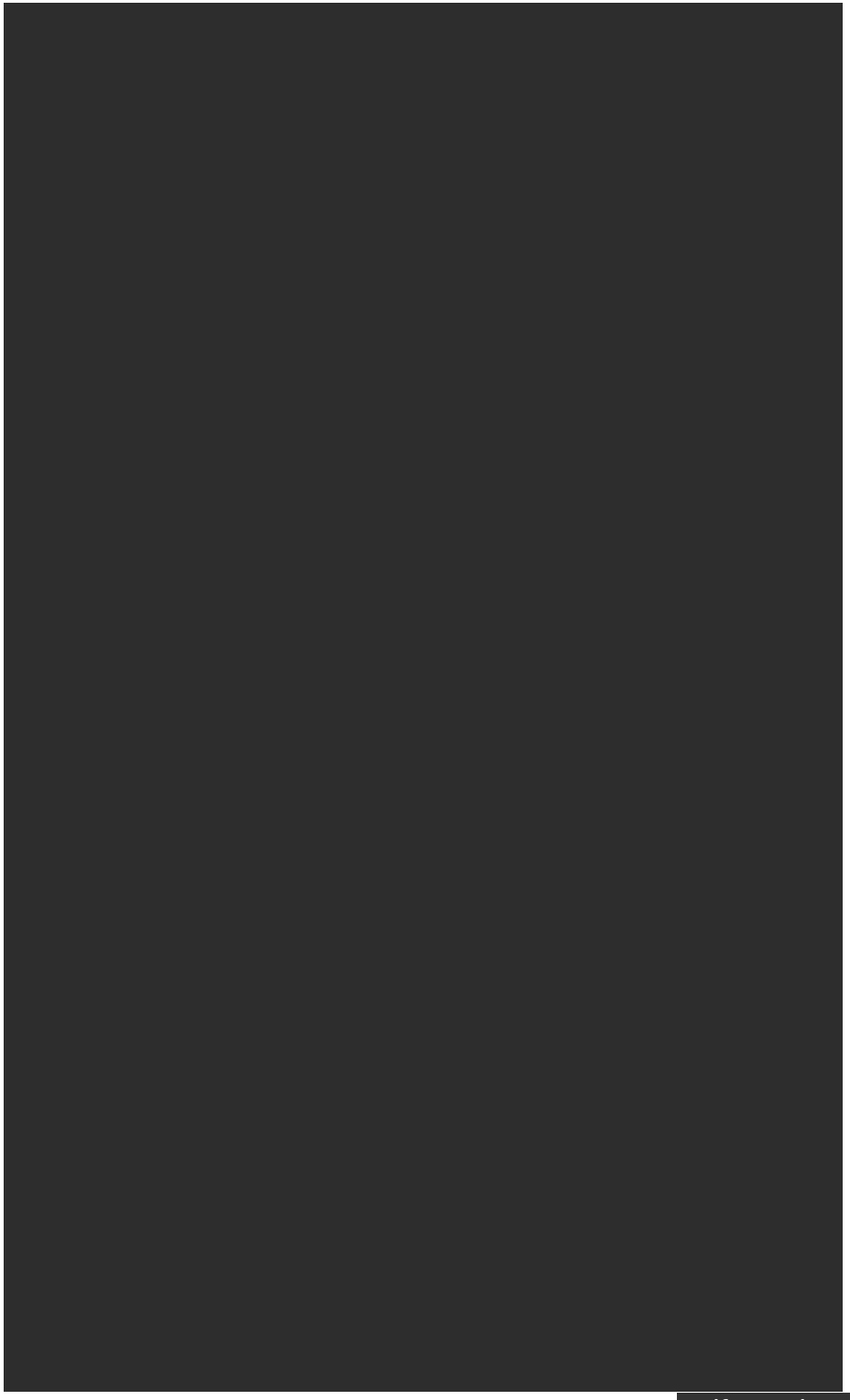
403: Forbidden

404: The room does not exist

Possible output fields

(Click here to show/hide)

Returns



Returns the data on the room queried. Details on returned data:

Get a specific room by offer ID

Endpoint

GET

/api/messages/v3/{company_user_id}/rooms/offers/{offer_id}.{format}

Gets the room information for the interview room with the offerId passed on the URL.

Required key permissions

Gets the rooms for the current user and organization

Arguments

company_user_id :	**required, string** The company ID or team ID, or user ID. Use 'List companies' or 'List teams' API calls to get the team/company IDs.
offer_id:	**required, integer** The ID of the offer to retrieve the room for.
format:	**optional, string** **Default:** `json`

DEFINITION

EXAMPLE RESPONSE

```
{
  'auth_user': {
  # ..
  },
  'server_time': 123456790,
  'room': {
    'roomId': 'room_12345',
    'roomName': 'Room name',
    # ..
  }
}
```

Response format.

Parameters

onlyRoomId: **optional, boolean**

Valid values: `true, false`

Indicates whether or not to return only the ID of the matching room.

Error messages

401: Unauthorized

403: Forbidden

404: The room does not exist

Possible output fields

(Click here to show/hide)

Returns

Returns the information on the offer ID queried. Details on returned data:

Get a specific room by application ID

Endpoint

GET

/api/messages/v3/{company_user_id}/rooms/applications/{application_id}.{format}

Gets the room information for the interview room with the applicationId passed on the URL.

Required key permissions

DEFINITION

EXAMPLE RESPONSE

```
{
  'auth_user': {
  # ..
  },
  'server_time': 123456790,
  'room': {
    'roomId': 'room_12345',
    'roomName': 'Room name',
    # ..
  }
```

Gets the rooms for the current user and organization

Arguments

company_user_id :	**required, string** The company ID or team ID, or user ID. Use 'List companies' or 'List teams' API calls to get the team/company IDs.
application_id:	**required, integer** The ID of the application to retrieve the room for.
format:	**optional, string** **Default:** `json` Response format.

Parameters

onlyRoomId:	**optional, boolean** **Valid values:** `true, false` Indicates whether or not to return only the ID of the matching room.

Error messages

401:	Unauthorized
403:	Forbidden
404:	The room does not exist

Possible output fields

(Click here to show/hide)

Returns

Returns the information on the application ID queried.

Details on returned data:

```
    }
```

Get a specific room by contract ID

Endpoint

GET

/api/messages/v3/{company_user_id}/rooms/contracts/{contract_id}.{format}

Gets the room information for the interview room with the contractId passed on the URL.

Required key permissions

Gets the rooms for the current user and organization

Arguments

company_user_id :	**required, string** The company ID or team ID, or user ID. Use 'List companies' or 'List teams' API calls to get the team/company IDs.
contract_id:	**required, integer** The ID of the contract to retrieve the room for.
format:	**optional, string** **Default:** `json` Response format.

Parameters

DEFINITION

EXAMPLE RESPONSE

```
{
  'auth_user': {
  # ..
  },
  'server_time': 123456790,
  'room': {
    'roomId': 'room_12345',
    'roomName': 'Room name',
    # ..
  }
}
```

onlyRoomId: optional, boolean

Valid values: `true, false`

Indicates whether or not to return only the ID of the matching room.

Error messages

401: Unauthorized

403: Forbidden

404: The room does not exist

Possible output fields

(Click here to show/hide)

Returns

Returns the information on the contract ID queried. Details on returned data:

Create a new room

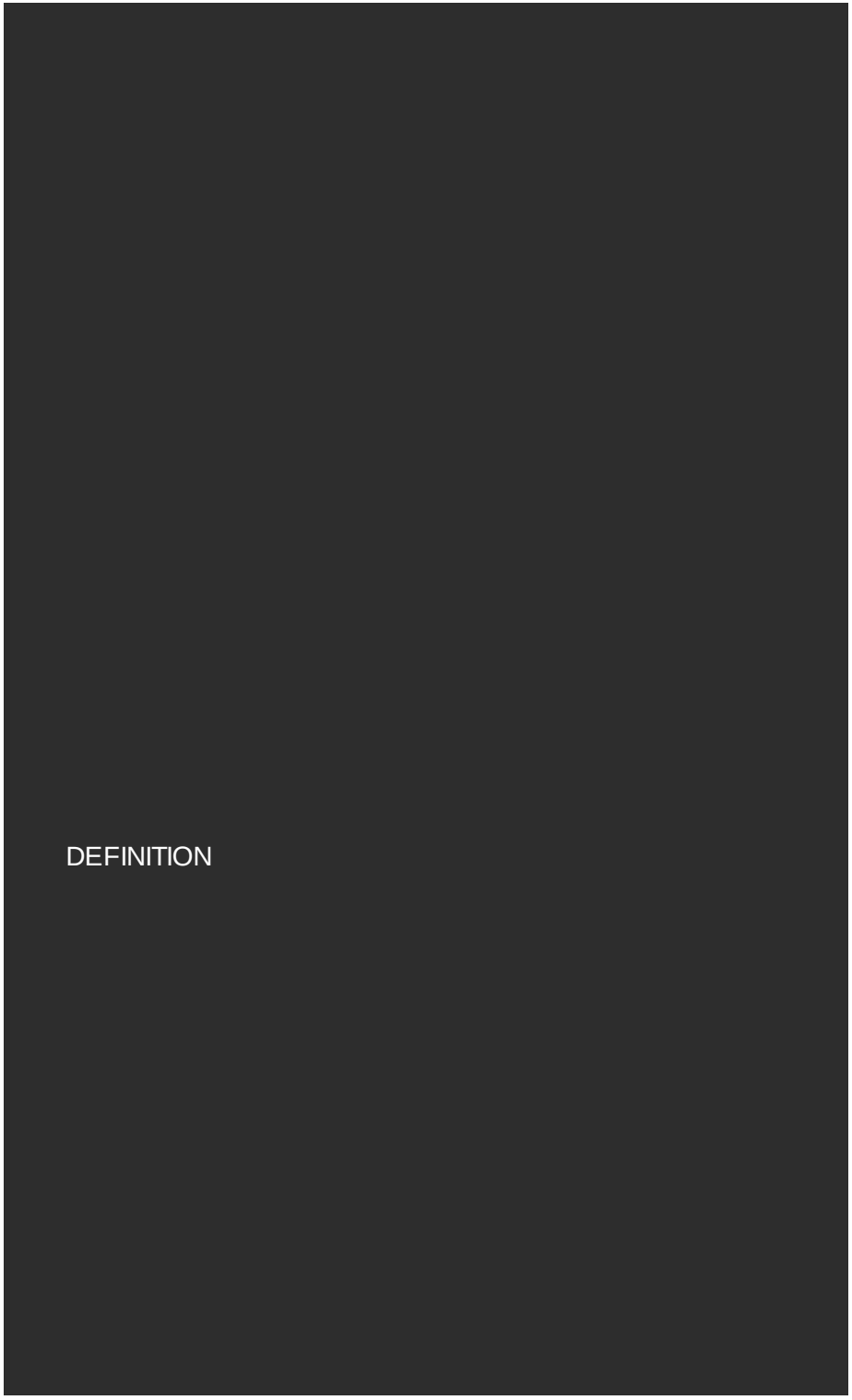
Warning

All parameters must be wrapped into json object, e.g. *{"roomName":"Group Project","roomType":"GROUP", ... }*, and sent as a separate parameter called *room*.

Endpoint

POST /api/messages/v3/{company_user_id}/rooms.{format}

Creates a new room. Note that 1-1 rooms must be passed in a single user in the users array. Both users are created



with `admin` role in the room and the role you pass is ignored (there is no owner).

Required key permissions

Create a new room, update an existent one or post a message in the room

Arguments

company_user_id **required, string**

: The company ID or team ID, or user ID. Use 'List companies' or 'List teams' API calls to get the team/company IDs.

format: **optional, string**

Default: `json`

Response format.

Parameters

roomName: **required, string**

The name of the room.

roomType: **optional, string**

Valid values: `GROUP, ONE_ON_ONE`

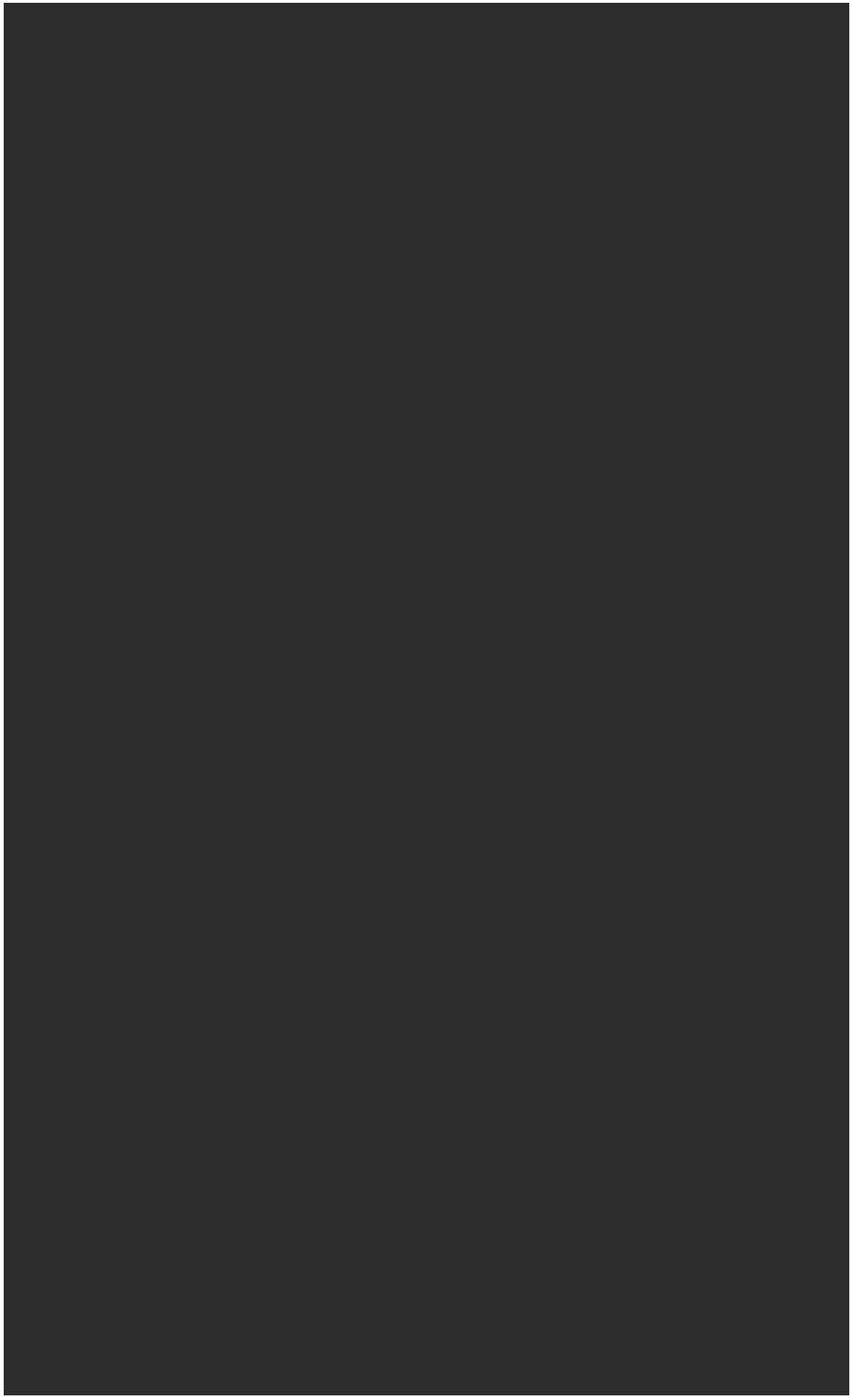
The room type.

topic: **optional, string**

The topic of the room. Note that 1:1 rooms do not have topics.

context: **optional, string**

Any contextual information related to the room.



users: **optional, array of json objects**

The initial users that should be in the room. For one-on-one rooms, 2 users are required to create a room.

role: **optional, string**

Valid values: `owner, admin, participant`

Only the current owner can change the room to contain a new owner. Admins can change the room name and topic, and invite new users. This field is a part of `users` field, e.g. `users { {userId: '~cipher', orgId: 'mycompany', role: 'admin', firstName: 'julian'}, {userId: '~cipher2', orgId: 'mycompany', role: 'owner', firstName: 'maksym'} }`

objectReferences: **optional, array of json objects**

Object references associated with the story.

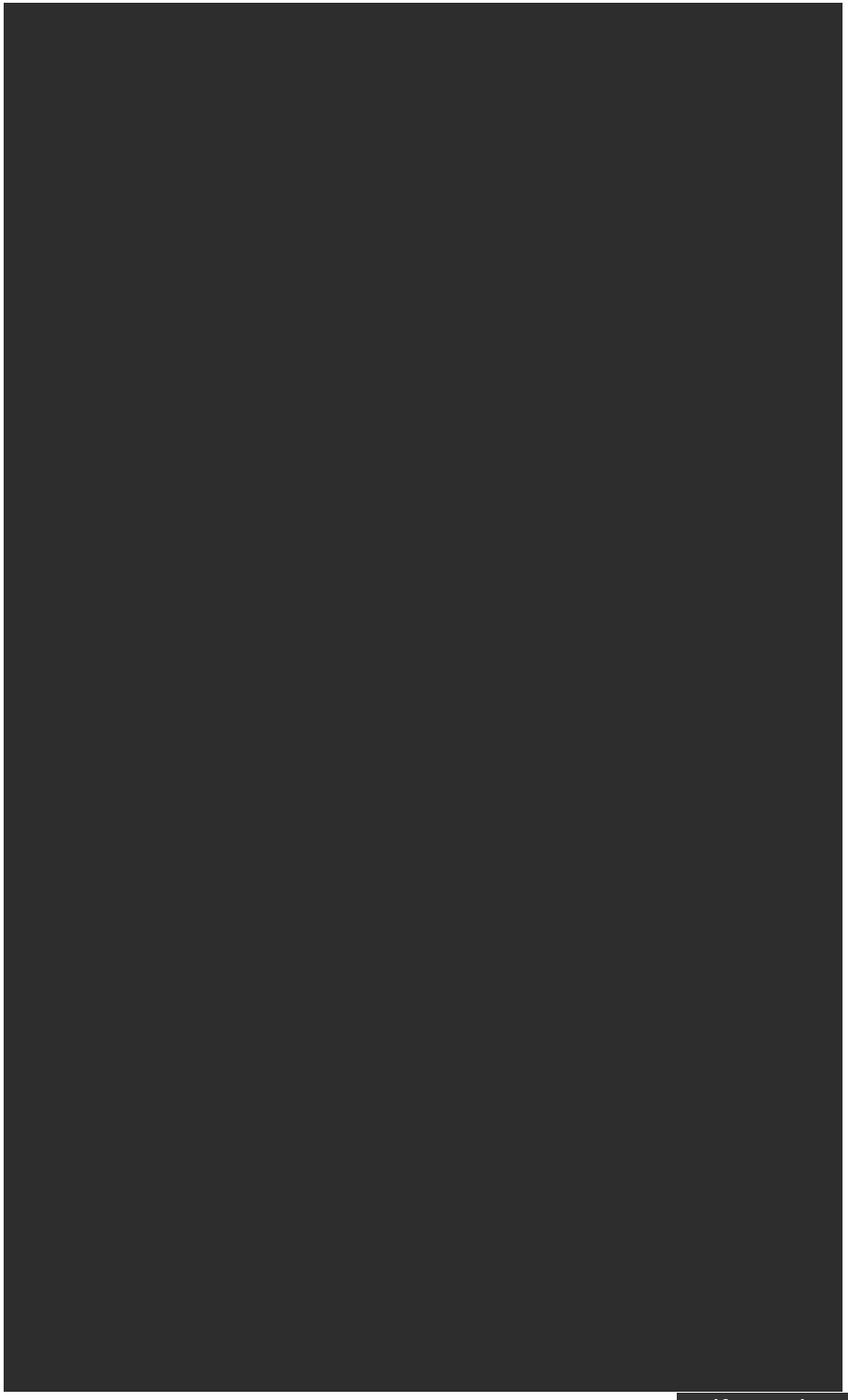
jobApplicationId: **optional, string**

The ID of the job application associated with the room, if applicable.

offerIds: **optional, string**

The IDs of any offers associated with the room.

contractId: **optional, string**



The ID of the job contract (parent assignment) associated with the room, if applicable.

Error messages

201:	Created
401:	Unauthorized
403:	Forbidden
400:	Bad Request

Returns

Returns HTTP code 201. Includes a link to the room in the Location header and the room ID in the response body, if creation was successful.

Send a message to a room

Warning

All parameters must be wrapped into json object, e.g. `{“message”: “a message”, ...}`, and sent as a separate parameter called `story`.

Endpoint

POST

/api/messages/v3/{company_user_id}/rooms/{room_id}/stories.{format}

Adds a new story to the given room.

DEFINITION

Required key permissions

Create a new room, update an existent one or post a message in the room

Arguments

company_user_id **required, string**

: The company ID or team ID, or user ID. Use 'List companies' or 'List teams' API calls to get the team/company IDs.

room_id: **required, string**

The ID of the room.

format: **optional, string**

Default: `json`

Response format.

Parameters

message: **optional, string**

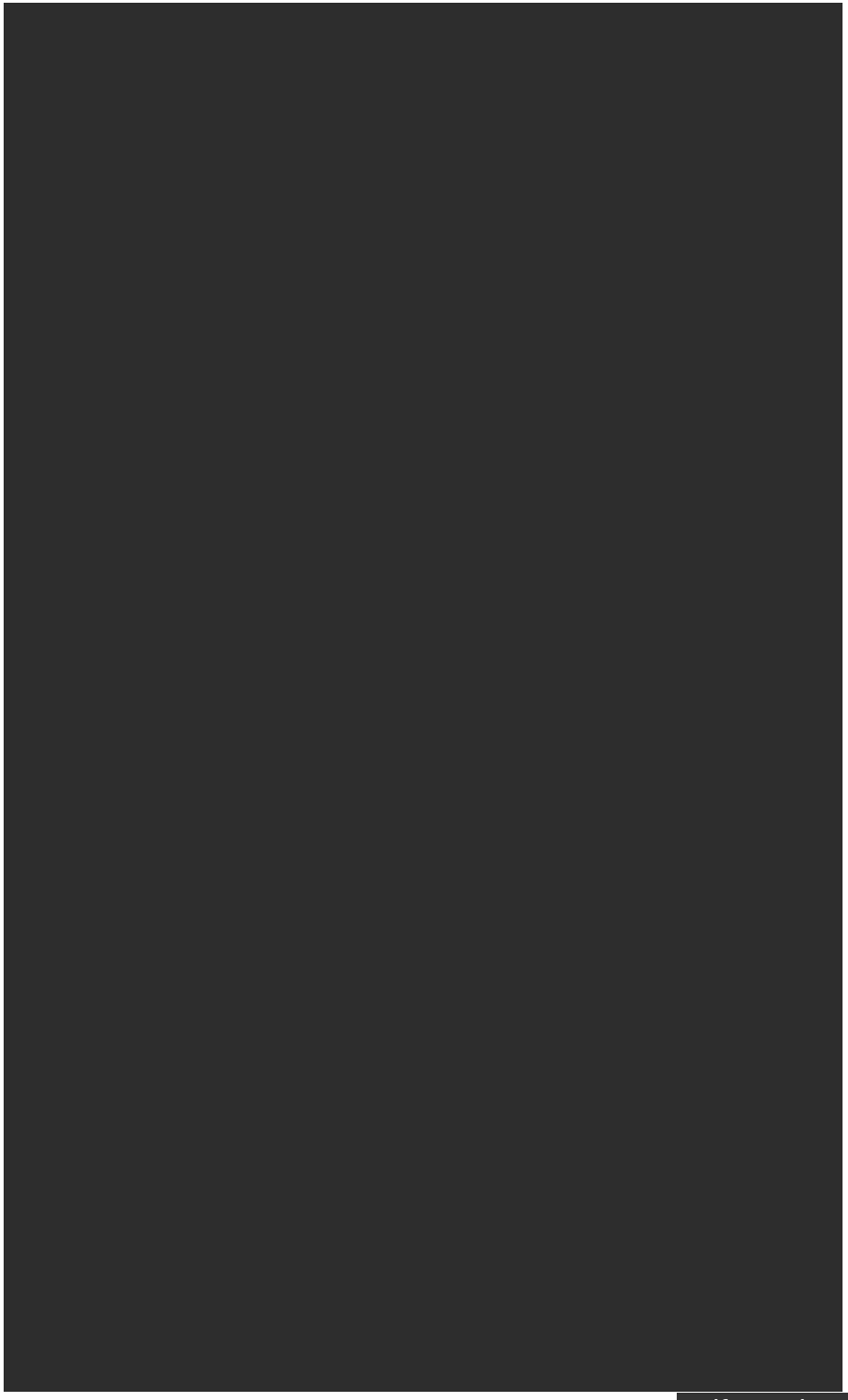
The custom user message associated with this Action and Object, if any. There is a configurable hard limit of 10KB on the length of the message.

userId: **optional, string**

The user ID of the author of the story. This is the user that performed the action.

orgId: **optional, string**

The org ID of the author of the story.



If a `userId` is provided, an `orgId` must be provided too.

Error messages

401: Unauthorized

403: Forbidden

Returns

Returns HTTP code 201.

Update a room settings

Warning

All parameters must be wrapped into json object, e.g. `{“isFavorite”: “true”}`, and sent as a separate parameter called `update_request`.

Endpoint

PUT

/api/messages/v3/{company_user_id}/rooms/{room_id}/users/{user_id}.{format}

Update a room settings.

Required key permissions

Create a new room, update an existent one or post a message in the room

Arguments

DEFINITION

company_user_id	**required, string**
:	The company ID or team ID, or user ID. Use 'List companies' or 'List teams' API calls to get the team/company IDs.
user_id:	**required, string**
	The ID of the user.
format:	**optional, string**
	Default: `json`
	Response format.

Parameters

lastReadTimesta	**optional, string**
mp:	Timestamp of the last time the user accessed the room.
isFavorite:	**optional, boolean**
	Valid values: `true, false`
	Indicates whether or not the user marked the room as a favorite.
isHidden:	**optional, string**
	Valid values: `true, false`
	Indicates whether or not the room is hidden from the user's left-nav.
role:	**optional, string**
	Valid values: `owner, admin, participant`
	The role of the user in the room.

Error messages

401:	Unauthorized

403: Forbidden

Returns

Returns HTTP code 200.

Archive or rename a room

Warning

All parameters must be wrapped into json object, e.g. `{“isReadOnly”: “true”}`, and sent as a separate parameter called `metadata`.

Endpoint

PUT

/api/messages/v3/{company_user_id}/rooms/{room_id}.{format}

Updates the metadata of an existing room and archives it.

Required key permissions

Create a new room, update an existent one or post a message in the room

Arguments

company_user_id **required, string**

: The company ID or team ID, or user ID. Use 'List companies' or 'List teams' API calls to get the

DEFINITION

team/company IDs.

room_id: **required, string**

The ID of the room.

format: **optional, string**

Default: `json`

Response format.

Parameters

roomName: **optional, string**

The new display name of the room, if the room is being renamed.

topic: **optional, string**

The topic of the conversation currently going on in the room, if any.

isReadOnly: **optional, string**

Marks the room read-only (archived). Setting this to true will append the text ` (archived)` to the room name. Setting both this argument and the `roomName` in the same request is prohibited.

Error messages

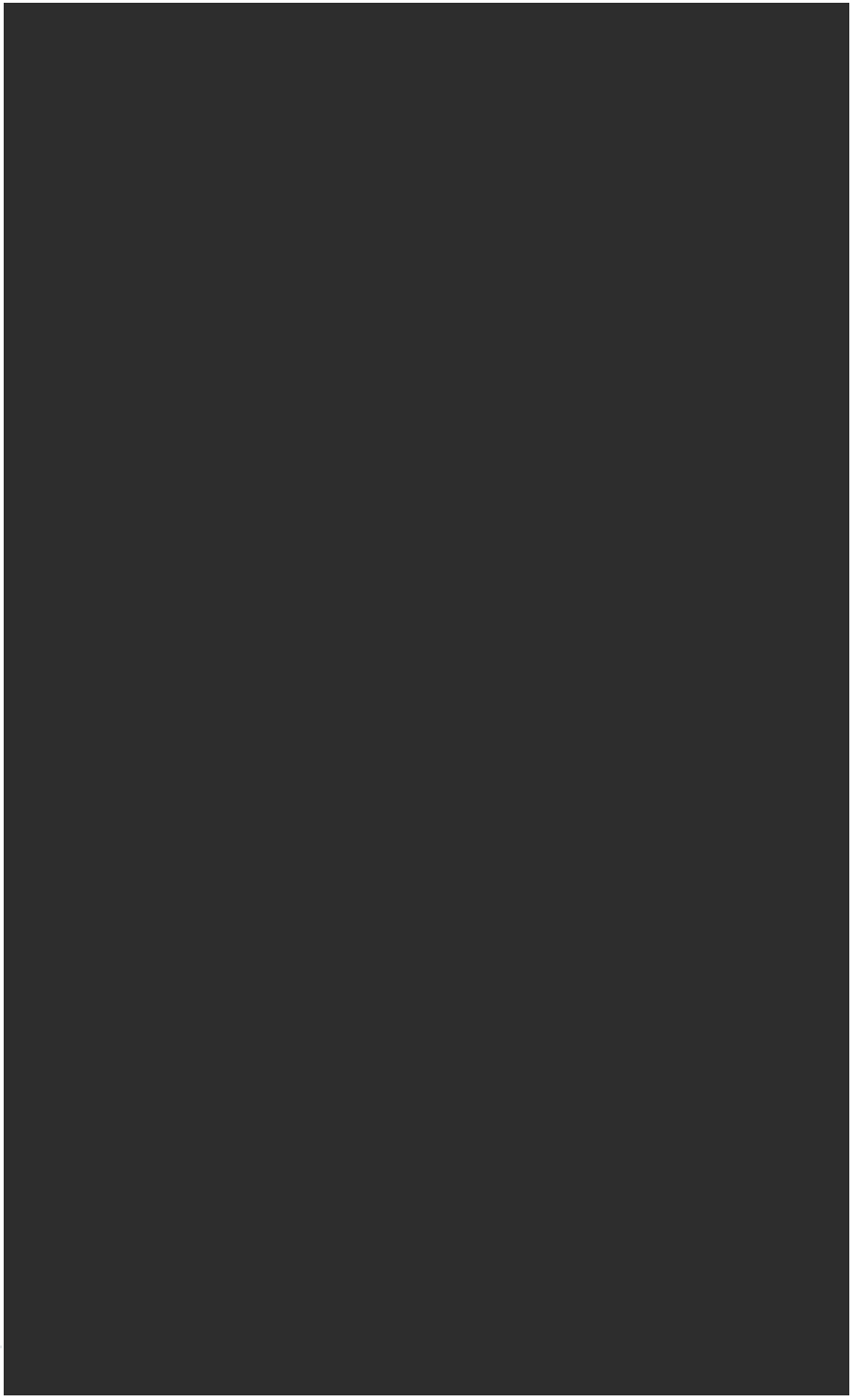
401: Unauthorized

403: Forbidden

404: Not Found

Returns

Returns HTTP code 200.



Activities

This section describes a set of resources that allow you to manage *activities* – Upwork task management system. The term “activity” is used with the meaning of “task” throughout the following reference documentation.

Activities API resources allow you to assign specific *activities* to freelancers. These resources require authentication and require the user to be a member of the referenced company or team, and have **hiring** permissions.

If a company has no teams, use the `company_id` as the `team_id`.

You can find `company_id` via Companies & Teams API, List teams in company. The value you need is returned in the `parent_team__id` field.

List team activities

Endpoint

GET

/api/otask/v1/tasks/companies/{company_id}/teams/{team_id}/tasks.{format}

This call returns a flat list (with no company/user description) of all activities under the specified team.

DEFINITION

```
activities.getList(company, team,
callback);
```

EXAMPLE REQUEST

```
var Team = require('upwork-
api/lib/routers/activities/team.js').Team
```

Required key permissions

View task codes

Arguments

company_id: **required, string**

The company ID. Use the `parent_team__id` value from the `Get teams` API call.

team_id: **required, string**

The team ID.

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Parameters

search: **optional, string**

Search term to filter tasks by code and description.

page: **optional, string**

Pagination, formed as `$offset;$count`. Example: `page=50;50`.

Error messages

401: Unauthorized

403: No access to company

403: No access to team

Returns

```
;

var activities = new Team(api);
activities.getList('123abc', '567def',
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
[
  {
    'code': 'User task 01',
    'company_id': '567def',
    'created_time': '2014-07-17
10:54:30',
    'description': 'Descr',
    'engagements': {
      'engagement': '171318'
    },
    'level': 'team',
    'payment_verification_status':
'VERIFIED',
    'record_id': '65464',
    'status': 'active',
    'team_id': '123abc',
    'total_engagements': '1',
    'url': '',
    'user_id': ''
  },
  {
    # Another activity
  },
  # ...
]
```

Details on the returned data:

record_id: The record ID of the activities. Each activity has a unique ID (users have no control over this number and it should only be used as a reference).

company_id: This is the ID of the company.

user_id: The Upwork ID of the user.

code: This is specified by the user. It could be used to reference a bug ID on a third-party system.

description: The text description contained within the activity.

URL: This is specified by the user. It can be used to link to a third-party bug tracking system or any related URL.

List activities for specific engagement

Endpoint

GET

/api/tasks/v2/tasks/contracts/{engagement_ref}.{format}

Authorized user must have hiring permissions in order to use this API call.

DEFINITION

```
activities.getSpecific(engagementRef,
callback);
```

EXAMPLE REQUEST

Required key permissions

View task codes

Arguments

engagement_ref: **required, integer**

Engagement reference ID. You can get it using 'List engagemnets' API call. Example: `1234`.

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Parameters

page: **optional, string**

Pagination, formed as `$offset;$count`. Example: `page=50;50`.

Error messages

401: Unauthorized

403: No access to company with this contract

Returns

Returns the list of activities that are currently assigned to the specified engagement.

```
activities.getSpecific(engagementRef,
callback);
```

EXAMPLE REQUEST

```
var Engagement = require('upwork-
api/lib/routers/activities/engagement.js'
).Engagement;

var activities = new Engagement(api);
activities.getSpecific('1234',
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'server_time': '1406027484',
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'task_sets': {
    'task_set': {
      'required': '1',
      'type': 'otask',
      'tasks': {
        'task': [
          {
            'description': 'Test task
01',
            'url': '',
            'code': 'test01',
            'id':
'{type=otask,cny=mytestcompany:mysecondsu
bteam,code=test01,team=mytestcompany:myse
condsubteam,level=team}'
          },
```

Get team activity by code

Endpoint

GET

/api/otask/v1/tasks/companies/{company_id}/teams/{team_id}/tasks/{code}.{format}

This call returns details on a single activity or a set of activities within a team.

Required key permissions

View task codes

Arguments

company_id:	**required, string** The company ID. Use the `parent_team__id` value from the 'Get teams' API call.
team_id:	**required, string**

```
            {
              # Another activity
            },
            # ...
          ]
        },
        'hash':
'59c31ed3d95d378740fa65218d8de7ae'
      }
    }
}
```

DEFINITION

```
activities.getSpecificList(
  company, team, code, callback);
```

EXAMPLE REQUEST

```
var Team = require('upwork-
api/lib/routers/activities/team.js').Team
;

var activities = new Team(api);
activities.getSpecificList('123abc',
'567def', 'User task 01', function(error,
data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

In case of multiple activities requested, a list will be returned.

The team ID.

code: **required, string**

The specific task code. It can be the list of codes, separated by ';' (semicolon). Example: `code=mycode01;mycode02;XYZ`.

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Error messages

401:	Unauthorized
403:	No access to company
403:	No access to team

Returns

In case of multiple activities requested, a list will be returned.

Details on the returned data:

record_id:	The record ID of the activities. Each activity has a unique ID (users have no control over this number and it should only be used as a reference).
company_id:	This is the ID of the company.
user_id:	The Upwork ID of the user.

```
{
  'code': 'User task 01',
  'company_id': '123abc',
  'created_time': '2014-07-17 10:54:30',
  'description': 'Descr',
  'engagements': {
    'engagement': '171318'
  },
  'level': 'team',
  'payment_verification_status':
'VERIFIED',
  'record_id': '65464',
  'status': 'active',
  'team_id': '678def',
  'url': '',
  'user_id': ''
}
```

code: This is specified by the user. It could be used to reference a bug ID on a third-party system.

description: The text description contained within the activity.

URL: This is specified by the user. It can be used to link to a third-party bug tracking system or any related URL.

Create activity at team level

Endpoint

POST

/api/otask/v1/tasks/companies/{company_id}/teams/{team_id}/tasks.{format}

A user needs to have hiring manager privileges within the team in order to create an activity at team level.

Required key permissions

Modify task codes

Arguments

company_id: **required, string**

The company ID. Use the `parent_team__id` value from the 'Get teams' API call.

DEFINITION

```
activities.addActivity(
  company, team, params, callback);
```

EXAMPLE REQUEST

```
var Team = require('upwork-
api/lib/routers/activities/team.js').Team
;

var activities = new Team(api);
var params = {
  'code': 'Team task 01',
  'description': 'Desc',
  'all_in_company': '1'
};
activities.addActivity('123abc',
'678def', params, function(error, data) {
  console.log(data);
});
```

team_id: **required, string**

The team ID.

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Parameters

code: **required, string**

An activity tracking code (this defines the activity tracking code).

description: **required, string**

The description of the code being added.

url: **optional, string**

The location where more info on the code can be found (the URL can be used to direct the user to more info about the code being entered).

engagements: **optional, string**

A list of engagements that are to be assigned to the created activity. It can be a single engagement ID, or a semicolon (';') separated list of IDs.

all_in_company: **optional, integer**

Valid values: `0, 1`

If set to `1`, then the created activity is assigned to all engagements that exist in the company at the moment.

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': '1405695093',
  'tasks': {
    'code': 'Team task 01',
    'company__reference': '377329',
    'company_id': '123abc',
    'created_time': '',
    'description': 'Desc',
    'engagements': '',
    'record_id': '65491',
    'reference': '65491',
    'status': '',
    'team__reference': '377329',
    'team_id': '678def',
    'url': '',
    'user__reference': '',
    'user_id': ''
  }
}
```

Error messages

401:	Unauthorized
403:	No access to company
403:	No access to team

Returns

If the call is successful, the server returns a 200 OK message. If the call fails a 403 error message is returned.

Update activity at team level

If you want to update an activity for the whole company, just use the `company_id` as the `team_id`.

Endpoint

PUT

/api/otask/v1/tasks/companies/{company_id}/teams/{team_id}/tasks/{code}.{format}

The description and URL of the activity can be updated.

Required key permissions

Modify task codes

Arguments

company_id:	**required, string** The company ID. Use the `parent_team__id` value from the 'Get teams' API call.

DEFINITION

```
activities.updateActivity(
  company, team, code, params, callback);
```

EXAMPLE REQUEST

```
var Team = require('upwork-
api/lib/routers/activities/team.js').Team
;

var activities = new Team(api);
var params = {
  'description': 'Desc updated',
  'all_in_company': '1'
};
activities.updateActivity('123abc',
'678def', 'Team task 01', params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

team_id: **required, string**

The team ID.

code: **required, string**

An activity tracking code being updated.

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Parameters

description: **required, string**

The description of the code being added.

url: **optional, string**

The location where more info on the code can be found (the URL can be used to direct the user to more info about the code being entered).

engagements: **optional, string**

A list of engagements that are to be assigned to the created activity. It can be a single engagement ID, or a semicolon (';') separated list of IDs.

all_in_company: **optional, integer**

Valid values: `0, 1`

If set to `1`, then the created activity is assigned to all engagements that

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': '1405746487',
  'tasks': {
    'message': 'updated'
  }
}
```

exist in the company at the moment.

Error messages

401:	Unauthorized
403:	No access to company
403:	No access to team

Returns

If the call is successful, the server returns a 200 OK message. If the call fails a 403 error message is returned.

Assign engagements to the list of activities

Endpoint

PUT

/api/otask/v1/tasks/companies/{company_id}/teams/{team_id}/engagements/{engagement}/tasks.{format}

Activity appears in freelancer's team client only if his engagement is assigned to the activity and activities are activated for the ongoing contract. This call overrides assigned engagements for the given activities. For example, if you pass empty `task_codes` or just omit this parameter, freelancer's engagement is unassigned from all the activities it is assigned to.

Required key permissions

Modify task codes

DEFINITION

```
activities.assign(
  company, team, engagementRef,
  params, callback);
```

EXAMPLE REQUEST

```
var Engagement = require('upwork-
api/lib/routers/activities/engagement.js'
).Engagement;

var activities = new Engagement(api);
var params = {'tasks': 'Team task 01'};
activities.assign('123abc', '678def',
'12345', params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
```

Arguments

company_id: **required, string**

The company ID. Use the `parent_team__id` value from the 'Get teams' API call.

team_id: **required, string**

The team ID.

engagement: **required, string**

Engagement ID that is assigned/unassigned to the given list of activities.

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Parameters

tasks: **required, string**

Activity tracking code or list of codes separated by semicolon (';').

Error messages

401: Unauthorized

403: No access to company

403: No access to team

Returns

Returns 200 OK in case of success.

```
    'auth_user': {
      'first_name': 'John',
      'last_name': 'Johnson',
      'timezone': 'Asia/Omsk',
      'timezone_offset': '25200'
    },
    'server_time': '1405882462',
    'tasks': ''
}
```

Assign to specific engagement the list of activities

Endpoint

PUT

/api/tasks/v2/tasks/contracts/{engagement_ref}.{format}

Authorized user must have hiring permissions in order to use this API call. Activity appears in freelancer's team client only if his engagement is assigned to the activity and activities are activated for the ongoing contract. This call overrides assigned engagements for the given activities. For example, if you pass empty `task_codes` or just omit this parameter, freelancer's engagement is unassigned from all the activities it is assigned to.

Required key permissions

Modify task codes

Arguments

engagement_ref:	**required, integer** Engagement reference ID. You can get it using 'List engagemnets' API call. Example: `1234`.
format:	**optional, string** **Default:** `json` **Valid values:** `json` Response format.

DEFINITION

```
activities.assignToEngagement(engagementRef, params, callback);
```

EXAMPLE REQUEST

```
var Engagement = require('upwork-api/lib/routers/activities/engagement.js').Engagement;

var activities = new Engagement(api);
var params = {'tasks': 'Team task 01'};
activities.assignToEngagement('1234', params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'server_time': '1406027484',
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'message': 'assigned'
}
```

Parameters

tasks: **required, string**

Activity tracking code or list of codes separated by semicolon (';').

Error messages

401: Unauthorized

403: No access to company with this contract

Returns

Returns 200 OK in case of success.

Archive activities

Endpoint

PUT

/api/otask/v1/tasks/companies/{company_id}/teams/{team_id}/archive/{code}.{format}

If you want to archive activity for company, use the `company_id` value for `team_id` as well.

Required key permissions

Modify task codes

Arguments

company_id: **required, string**

The company ID. Use the `parent_team__id` value from the 'Get

DEFINITION

```
activities.archiveActivity(
  company, team, code, callback);
```

EXAMPLE REQUEST

```
var Team = require('upwork-
api/lib/routers/activities/team.js').Team
;

var activities = new Team(api);
activities.archiveActivity('123abc',
'678def', 'Team task 01', function(error,
data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

teams' API call.

team_id: **required, string**

The team ID.

code: **required, string**

An activity tracking code. It can be the list of codes, separated by ';' (semicolon).

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Error messages

401:	Unauthorized
403:	No access to company
403:	No access to team

Returns

Returns 200 OK in case of success.

Unarchive activities

Endpoint

PUT

/api/otask/v1/tasks/companies/{company_id}/teams/{team_id}/unarchive/{code}.{format}

If you want to unarchive activity for company, use the `company_id` value for `team_id` as well.

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': '1405883808',
  'tasks': '1'
}
```

DEFINITION

```
activities.unarchiveActivity(
  company, team, code, callback);
```

EXAMPLE REQUEST

```
var Team = require('upwork-
api/lib/routers/activities/team.js').Team
```

Required key permissions

Modify task codes

Arguments

company_id: **required, string**

The company ID. Use the `parent_team__id` value from the 'Get teams' API call.

team_id: **required, string**

The team ID.

code: **required, string**

An activity tracking code. It can be the list of codes, separated by ';' (semicolon).

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Error messages

401: Unauthorized

403: No access to company

403: No access to team

Returns

Returns 200 OK in case of success.

Update bulk of activities

```
;

var activities = new Team(api);
activities.unarchiveActivity('123abc',
'678def', 'Team task 01', function(error,
data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': '1405884666',
  'tasks': '1'
}
```

> **Note**
>
> Currently this API call gives no control over engagements assignment or archiving/unarchiving of activities. **Use it at your own risk.**

Endpoint

PUT

/api/otask/v1/tasks/companies/{company_id}/tasks/batch.{format}

Activities can be updated using a CSV file via the API. This process actually deletes the corresponding activities and replaces them with the newly created ones with the specified details. Note that authorized user has to have hiring permissions in order to use this API call.

Required key permissions

Modify task codes

Arguments

company_id:	**required, string** The company ID. Use the `parent_team__id` value from the 'Get teams' API call.
format:	**optional, string** **Default:** `xml` **Valid values:** `json, xml` Response format.

DEFINITION

```
activities.updateBatch(company, params,
callback);
```

EXAMPLE REQUEST

```
var Team = require('upwork-
api/lib/routers/activities/team.js').Team
;

var activities = new Team(api);
var companyId = '123abc';
var teamId = '678def';
var userId = 'john_freelancer';
var params = {
  'data': '"' + companyId + '","' +
teamId + '","' + userId + '","User task
01","Desc bulk",""<br>"' + companyId +
'","' + teamId + '","' + userId +
'","User task 02","Desc bulk",""'
};
activities.updateBatch(companyId, params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': '1405697909',
  'tasks': '2'
}
```

Parameters

data: **required, string**

Data for activities in CSV format but with `<br>` as line separator: `"{company_id}","{team_id}","{user_id}","{code}","{description}","{url}"`. Example: `"acmeinc","","","ABC","Project ABC","https://www.acmeinc.com/project/abc"<br>"acmeinc","acmeinc:dev","b42","123","Task 123","https://www.acmeinc.com/task/123"`. Note: the given data replaces the existing "acmeinc" company level records with the new "ABC" record and "acmeinc:dev":"b42" user level records with the new "123" record.

Error messages

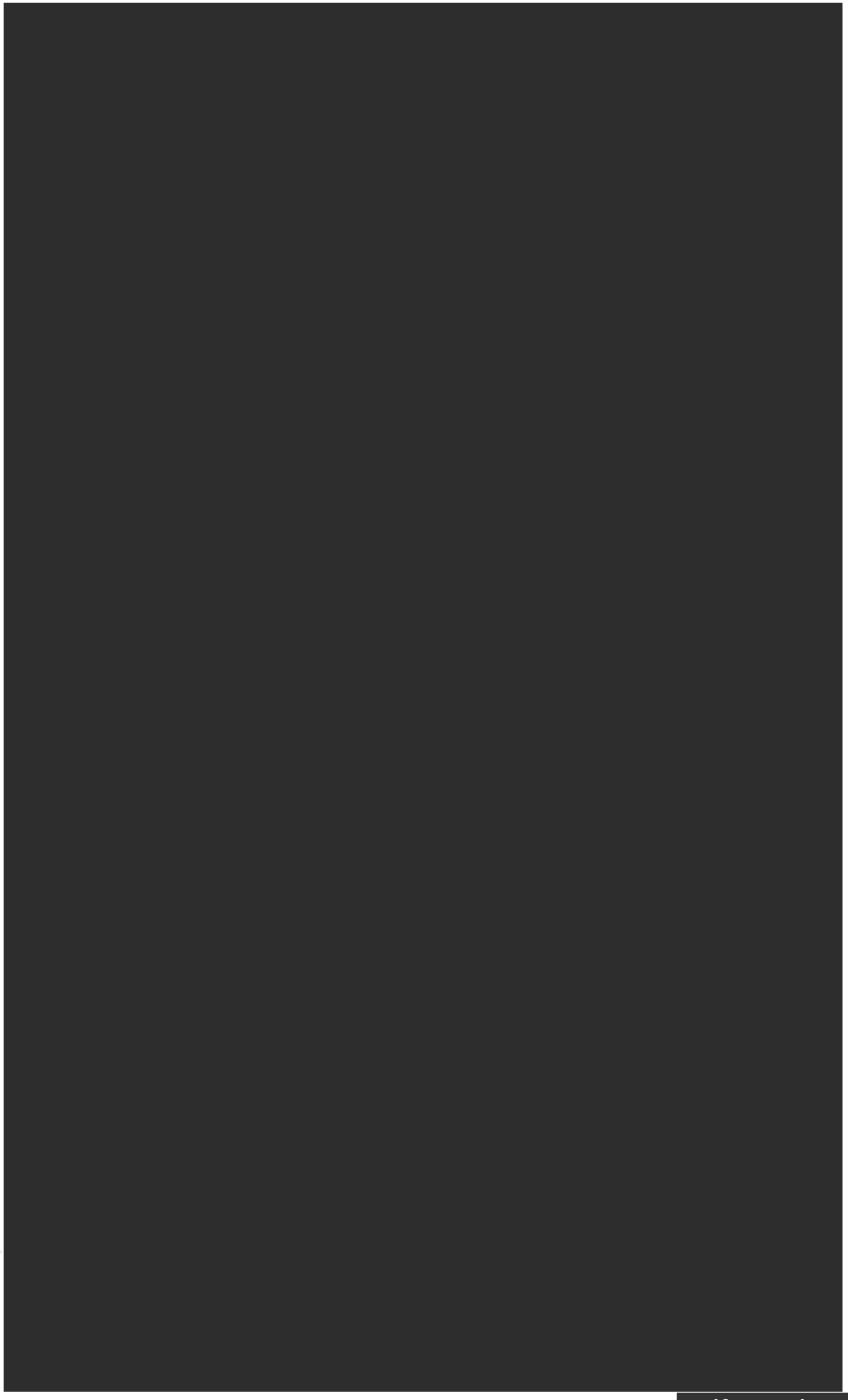
401: Unauthorized

403: No access to company

403: No access to team

Returns

If the call is successful, the server returns a 200 OK message. If the call fails a 403 error message is returned.



Snapshots

This section describes resources that allow you to get activity snapshots from a user account, in a company's team, at specific points in time. You can also update and delete snapshots. These resources can only be used with the teams the user has access to. An error will be returned if the currently authorized user does not have access to the teammate details being requested.

Get snapshot

Endpoint

GET

/api/team/v1/snapshots/{company_id}/{username}/{timestamp}.{format}

Required key permissions

View your workdiary

Arguments

company_id: **required, string**

The company ID.

username: **required, string**

The username of the target user account.

timestamp: **required, string**

The timestamp either in UTC according to ISO 8601 (yyyymmddTHHMMSSZ) or as a

DEFINITION

```
snapshots.get(
  company, username, ts, callback);
```

EXAMPLE REQUEST

```
var Snapshot = require('upwork-
api/lib/routers/snapshot.js').Snapshot;

var snapshots = new Snapshot(api);
snapshots.get('123abc',
'john_freelancer',
'20140101T090000Z,20140101T120000Z',
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'account_status': 'enabled',
  'active_window_title': '',
  'activity': '0',
```

(yyyymmddTHHMMSSZ) or as a UNIX timestamp (number of seconds after epoch). If absent, it defaults to the current time. If an exact timestamp match is not found, then the latest snapshot during the current and the previous 10 minute slots (i.e. up to 20 minutes past) is returned. If no activity is found in this time range, a snapshot with PRIVATE status is returned.

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Error messages

401: Unauthorized

403: Forbidden (user is not invited to specified company)

Returns

Details on the response:

snapshot: Snapshot detail container.

status: Status of the snapshot. Can be: `LOGIN`, `NORMAL`, `PRIVATE`, `EXIT`

time: The GMT Unix timestamp when the snapshot was taken


```
    'billing_status': 'non-
billed.disconnected',
    'cellts': '',
    'client_version': '',
    'company_id': 'unazcm8kj7ytdrwwgc-pkq',
    'computer_name': '',
    'digest': '',
    'keyboard_events_count': '0',
    'memo': '',
    'mouse_events_count': '0',
    'online_presence': '0,0,0,0...',
    'online_presence_img': 'https://...',
    'report24_img': 'https://...',
    'status': 'PRIVATE',
    'team_name': 'John Test oAuth API',
    'teamroom_api':
'/api/team/v1/teamrooms/unazcm8kj7ytdrwwg
c-pkq.json',
    'time': '1405919907',
    'timezone': 'Asia/Omsk',
    'uid': '',
    'user': {'archiving_time':
'1405555200',
    'creation_time': '1405434052',
    'first_name': 'John',
    'last_name': 'Panshine',
    'mail': 'john_freelancer@example.com',
    'messenger_id': None,
    'messenger_type': None,
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200',
    'uid': 'john_freelancer'},
    'user_id': 'john_freelancer',
    'workdiary_api':
'/api/team/v1/workdiaries/unazcm8kj7ytdrw
wgc-pkq/john_freelancer/20140721.json'
    }
```

billing_status: The billing status of the snapshot: `non-billed.disconnected`, `billed.disconnected`, `billed.connected`

activity: User activity. Aggregated number of keystrokes and mouse clicks.

online_presence: Indicates whether the user is online.

user: General details about the current user.

mouse_events_count: The number of mouse events associated with this snapshot.

company_id: Company ID associated with this snapshot.

timezone: User's time zone

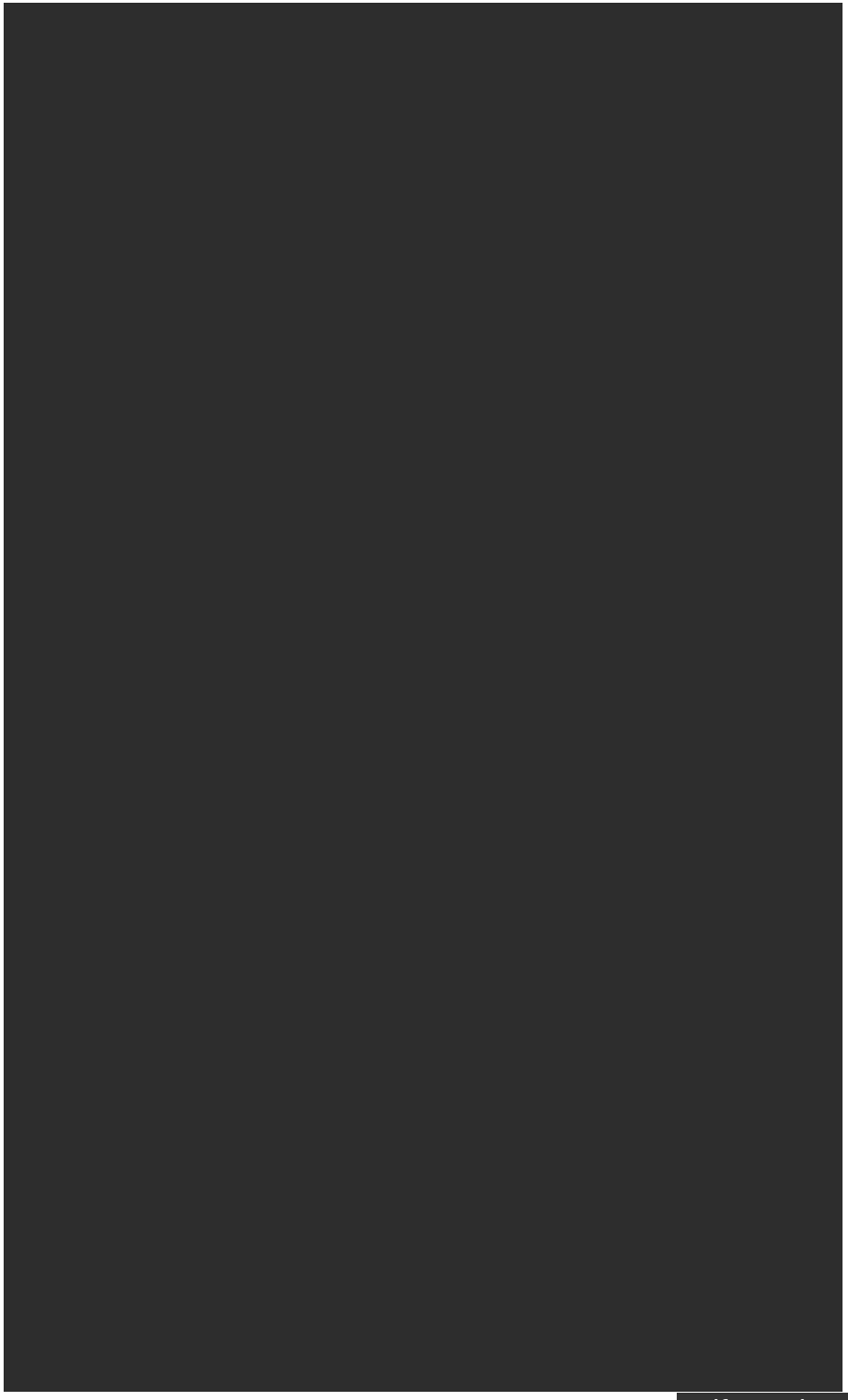
uid: The user ID.

keyboard_events_count: Number of keyboard events counted for this snapshot.

last_worked: The Unix timestamp indicating when the user last worked.

memo: Memo associated with the current time stamp.

active_window_title: The title of the active window when the snapshot was taken.



report24_img: URL to a graph that describes a user's activity over a 24hr period.

computer_name: The name of the computer where the snapshot was taken.

online_presence_img: URL for the default online user activity

user_id: User ID associated with the current snapshot

client_version: The Upwork Time Tracker version used to take the snapshot

Update snapshot memo

Endpoint

PUT

/api/team/v1/snapshots/{company_id}/{username}/{timestamp}.{format}

If the `timestamp` path parameter is not specified, the snapshot for 'now' is updated. Several snapshots can be updated as well, see the `timestamp` parameter description.

Required key permissions

Modify your workdiary

Arguments

DEFINITION

```
snapshots.update(
  company, username, ts, params,
  callback);
```

EXAMPLE REQUEST

```
var Snapshot = require('upwork-
api/lib/routers/snapshot.js').Snapshot;

var snapshots = new Snapshot(api);
params = {'memo': 'New memo'};
snapshots.update('123abc',
'john_freelancer',
'20140101T090000Z,20140101T120000Z',
params, function(error, data) {
  console.log(data);
});
```

company_id: **required, string**

The company ID.

username: **required, string**

The username of the target user account.

timestamp: **optional, string**

The timestamp either in UTC according to ISO 8601 (yyyymmddTHHMMSSZ) or as a UNIX timestamp (number of seconds after epoch). If absent, it defaults to the current time. More than one timestamps can be specified either as a range or as a list of values. For timestamp range use the comma character (','). Example: `20081205T090351Z,20081205T091853Z`. For list of timestamps, use the semicolon character (';'), example: `20081205T090351Z;20081405T090851Z;20081705T091853Z`.

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Parameters

memo: **required, string**

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405935643,
  'snapshot': 'added'
}
```

The Memo of the snapshot.

Error messages

401:	Unauthorized
403:	Forbidden (user is not invited to specified company)

Returns

If the snapshot is updated, a `200 OK` status is returned.

Delete snapshot

Endpoint

DELETE

/api/team/v1/snapshots/{company_id}/{username}/{timestamp}.{format}

One or several snapshots can be deleted, see the `timestamp` parameters description.

Required key permissions

Modify your workdiary

Arguments

company_id:	**required, string** The company ID.
username:	**required, string** The username of the target user account.
timestamp:	**optional, string**

DEFINITION

```
snapshots.delete(
  company, username, ts, callback);
```

EXAMPLE REQUEST

```
var Snapshot = require('upwork-
api/lib/routers/snapshot.js').Snapshot;

var snapshots = new Snapshot(api);
snapshots.delete('123abc',
'john_freelancer',
'20140101T090000Z,20140101T120000Z',
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
```

The timestamp either in UTC according to ISO 8601 (yyyymmddTHHMMSSZ) or as a UNIX timestamp (number of seconds after epoch). If absent, it defaults to the current time. More than one timestamps can be specified either as a range or as a list of values. For timestamp range use the comma character (','). Example: `20081205T090351Z,20081205T091853Z`. For list of timestamps, use the semicolon character (';'), example: `20081205T090351Z;20081405T090851Z;20081705T091853Z`.

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Error messages

401:	Unauthorized
403:	Forbidden (user is not invited to specified company)

Returns

If the snapshot is deleted, a `200 OK` message is returned.

```
        'timezone': 'Asia/Omsk',
        'timezone_offset': '25200'
      },
      'server_time': 1405935643,
      'snapshot': 'deleted'
    }
```

Get snapshot by Contract

Endpoint

GET

/api/team/v2/snapshots/contracts/{contract_id}/{timestamp}.{format}

Required key permissions

View your workdiary

Arguments

contract_id: **required, string**

The contract ID.

timestamp: **required, string**

The timestamp either in UTC according to ISO 8601 (yyyymmddTHHMMSSZ) or as a UNIX timestamp (number of seconds after epoch). If absent, it defaults to the current time. If an exact timestamp match is not found, then the latest snapshot during the current and the previous 10 minute slots (i.e. up to 20 minutes past) is returned. If no activity is found in this time range, a snapshot with PRIVATE status is returned.

format: **optional, string**

DEFINITION

```
snapshots.getByContract(contract_id, ts,
callback);
```

EXAMPLE REQUEST

```
var Snapshot = require('upwork-
api/lib/routers/snapshot.js').Snapshot;

var snapshots = new Snapshot(api);
snapshots.get_by_contract('1234',
'20140101T090000Z,20140101T120000Z',
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'account_status': 'enabled',
  'active_window_title': '',
  'activity': '0',
  'billing_status': 'non-
billed.disconnected',
  'cellts': '',
  'client_version': '',
  'company_id': 'unazcm8kj7ytdrwwgc-pkq',
  'computer_name': '',
  'digest': '',
  'keyboard_events_count': '0',
  'memo': '',
  'mouse_events_count': '0',
  'online_presence': '0,0,0,0...',
  'online_presence_img': 'https://...',
  'report24_img': 'https://...',
  'status': 'PRIVATE',
  'team_name': 'John Test oAuth API',
```

Default: `json`

Valid values: `json`

Response format.

Error messages

401:	Unauthorized
403:	Forbidden (user is not invited to specified company)

Possible output fields

(Click here to show/hide)

Returns

Details on the response:

snapshot:	Snapshot detail container.
status:	Status of the snapshot. Valid values are: `LOGIN`, `NORMAL`, `PRIVATE`, `EXIT`.
time:	The GMT Unix timestamp when the snapshot was taken.
billing_status:	The billing status of the snapshot: `non-billed.disconnected`, `billed.disconnected`, `billed.connected`.
activity:	User activity. Aggregated number of keystrokes and mouse clicks.
online_presence:	Indicates whether the user is


```
  'teamroom_api':
'/api/team/v1/teamrooms/unazcm8kj7ytdrwwg
c-pkq.json',
  'time': '1405919907',
  'timezone': 'Asia/Omsk',
  'uid': '',
  'user': {'archiving_time':
'1405555200',
  'creation_time': '1405434052',
  'first_name': 'John',
  'last_name': 'Panshine',
  'mail': 'john_freelancer@example.com',
  'messenger_id': None,
  'messenger_type': None,
  'timezone': 'Asia/Omsk',
  'timezone_offset': '25200',
  'uid': 'john_freelancer'},
  'user_id': 'john_freelancer',
  'workdiary_api':
'/api/team/v1/workdiaries/unazcm8kj7ytdrw
wgc-pkq/john_freelancer/20140721.json'
}
```

online.

user: General details about the current user.

mouse_events_count: The number of mouse events associated with this snapshot.

company_id: Company ID associated with this snapshot.

timezone: User’s time zone.

uid: The user ID.

keyboard_events_count: Number of keyboard events counted for this snapshot.

last_worked: The Unix timestamp indicating when the user last worked.

memo: Memo associated with the current time stamp.

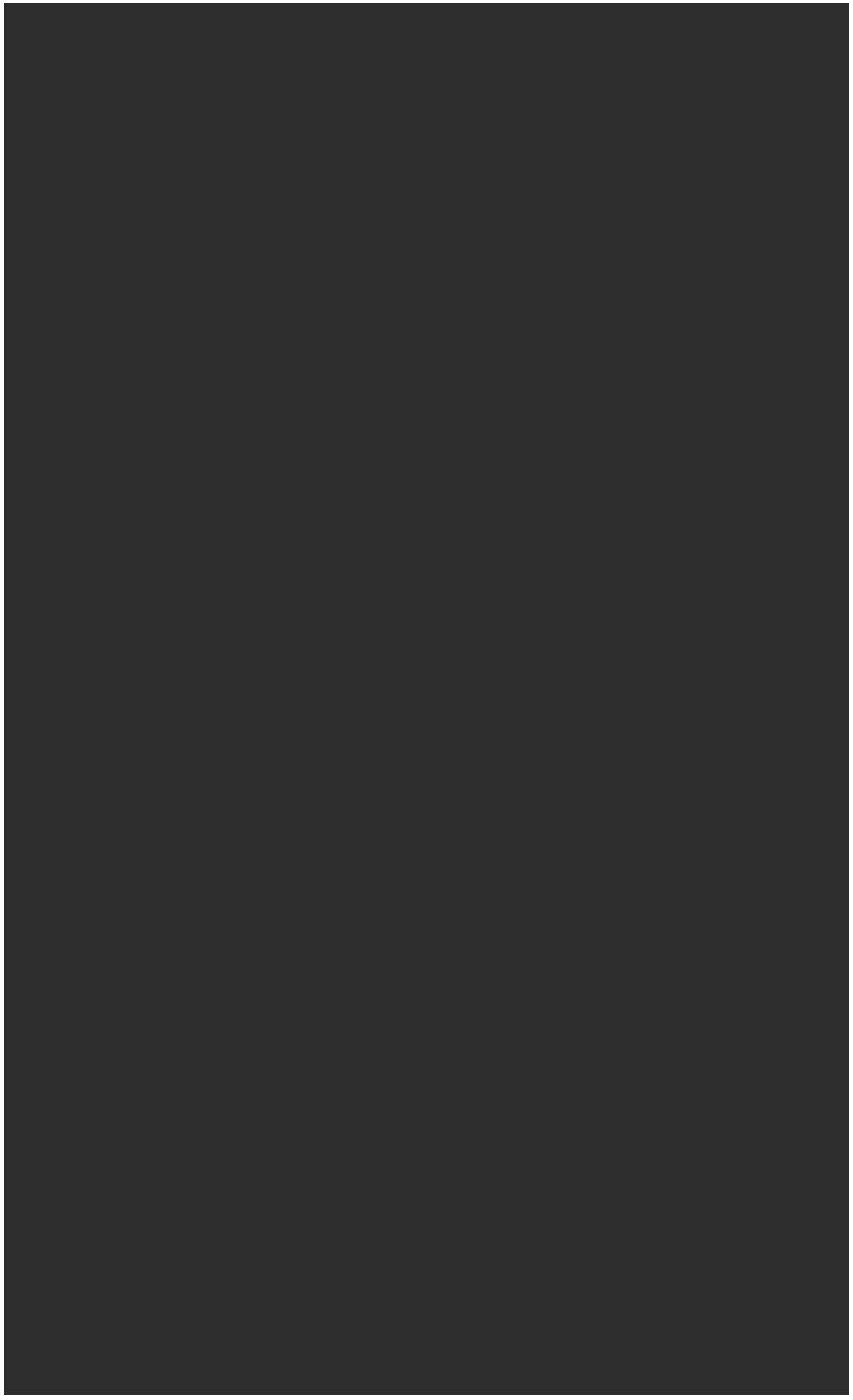
active_window_title: The title of the active window when the snapshot was taken.

report24_img: URL to a graph that describes a user’s activity over a 24hr period.

computer_name: The name of the computer where the snapshot was taken.

online_presence_img: URL for the default online user activity.

user_id: User ID associated with the



current snapshot.

client_version: The Upwork Time Tracker version used to take the snapshot.

Update snapshot memo by Contract

Endpoint

PUT

/api/team/v2/snapshots/contracts/{contract_id}/{timestamp}.{format}

If the `timestamp` path parameter is not specified, the snapshot for 'now' is updated. Several snapshots can be updated as well, see the `timestamp` parameter description.

Required key permissions

Modify your workdiary

Arguments

contract_id: **required, string**
The contract ID.

timestamp: **optional, string**
The timestamp either in UTC according to ISO 8601 (yyyymmddTHHMMSSZ) or as a UNIX timestamp (number of seconds after epoch). If absent, it defaults to

DEFINITION

```
snapshots.update_by_contract(
  contract_id, ts, params,
  callback);
```

EXAMPLE REQUEST

```
var Snapshot = require('upwork-
api/lib/routers/snapshot.js').Snapshot;

var snapshots = new Snapshot(api);
params = {'memo': 'New memo'};
snapshots.updateByContract('1234',
'20140101T090000Z,20140101T120000Z',
params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405935643,
  'snapshot': 'added'
}
```

the current time. More than one timestamps can be specified either as a range or as a list of values. For timestamp range use the comma character (','). Example: `20081205T090351Z,20081205T091853Z`. For list of timestamps, use the semicolon character (';'), example: `20081205T090351Z;20081405T090851Z;20081705T091853Z`.

format: **optional, string**

Default: `json`

Valid values: `json`

Response format.

Parameters

memo: **required, string**

The Memo of the snapshot.

Error messages

401: Unauthorized

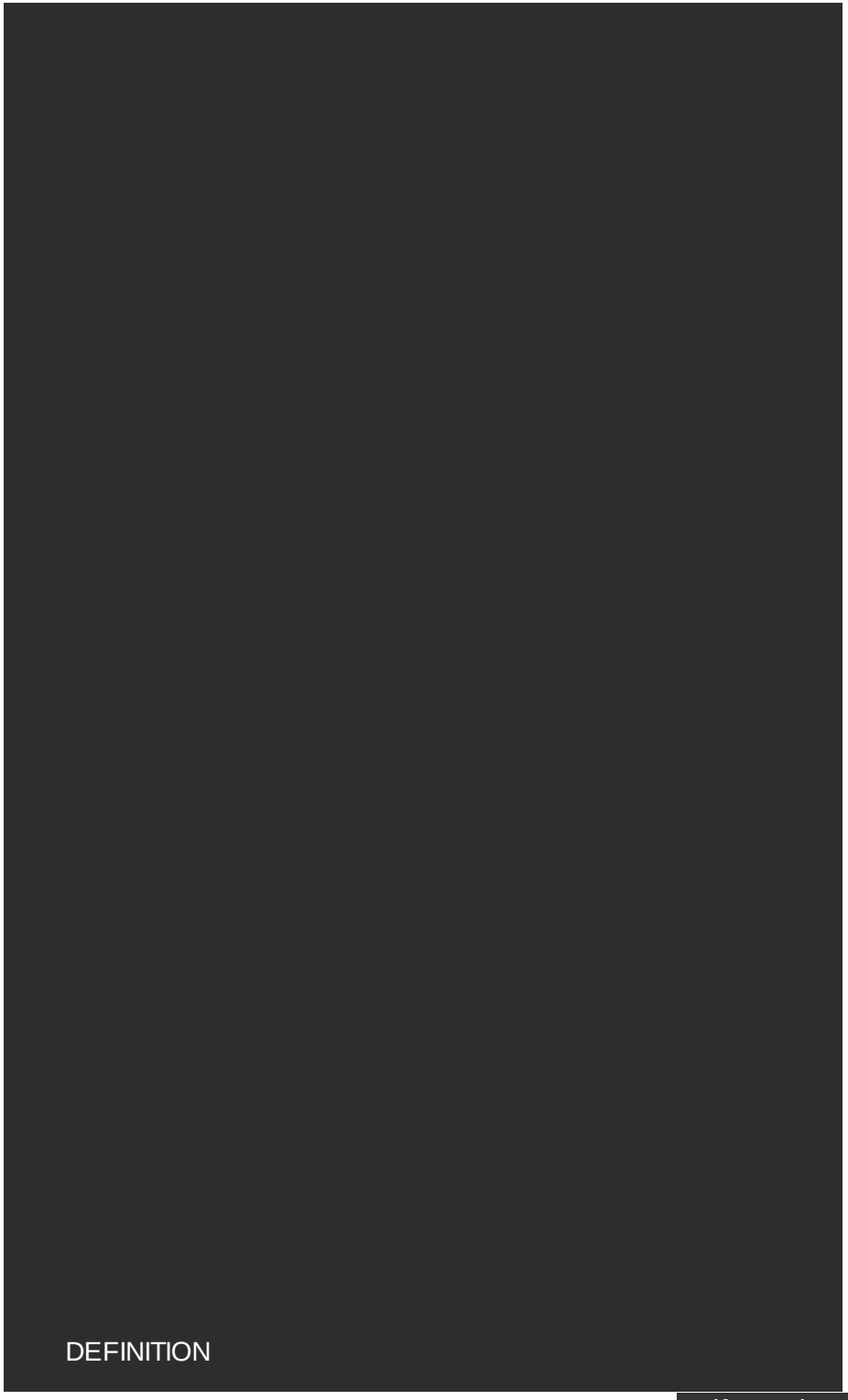
403: Forbidden (user is not invited to specified company)

Returns

If the snapshot is updated, a `200 OK` status is returned.

Delete snapshot by Contract

Endpoint



DELETE

/api/team/v2/snapshots/contracts/{contract_id}/{timestamp}.{format}

One or several snapshots can be deleted, see the `timestamp` parameters description.

Required key permissions

Modify your workdiary

Arguments

contract_id: **required, string**

The contract ID.

username: **required, string**

The username of the target user account.

timestamp: **optional, string**

The timestamp either in UTC according to ISO 8601 (yyyymmddTHHMMSSZ) or as a UNIX timestamp (number of seconds after epoch). If absent, it defaults to the current time. More than one timestamps can be specified either as a range or as a list of values. For timestamp range use the comma character (','). Example: `20081205T090351Z,20081205T0918

```
snapshots.deleteByContract(
  contract_id, ts, callback);
```

EXAMPLE REQUEST

```
var Snapshot = require('upwork-
api/lib/routers/snapshot.js').Snapshot;

var snapshots = new Snapshot(api);
snapshots.delete('1234',
'20140101T090000Z,20140101T120000Z',
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405935643,
  'snapshot': 'deleted'
}
```

53Z`. For list of timestamps, use the semicolon character (';'), example: `20081205T090351Z;20081405T090851Z;20081705T091853Z`.

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Error messages

401: Unauthorized

403: Forbidden (user is not invited to specified company)

Returns

If the snapshot is deleted, a `200 OK` message is returned.

Reports

This section describes resources that use Google Visualization APIs query language to run detailed reports on a user's team, company or a specific freelancer.

In order to use these resources it is necessary to understand the data model and the supported Google query language elements.

For more details around how to construct data parameters see Google documentation.

Below you will find the detailed description of what fields are available for constructing GDS queries.

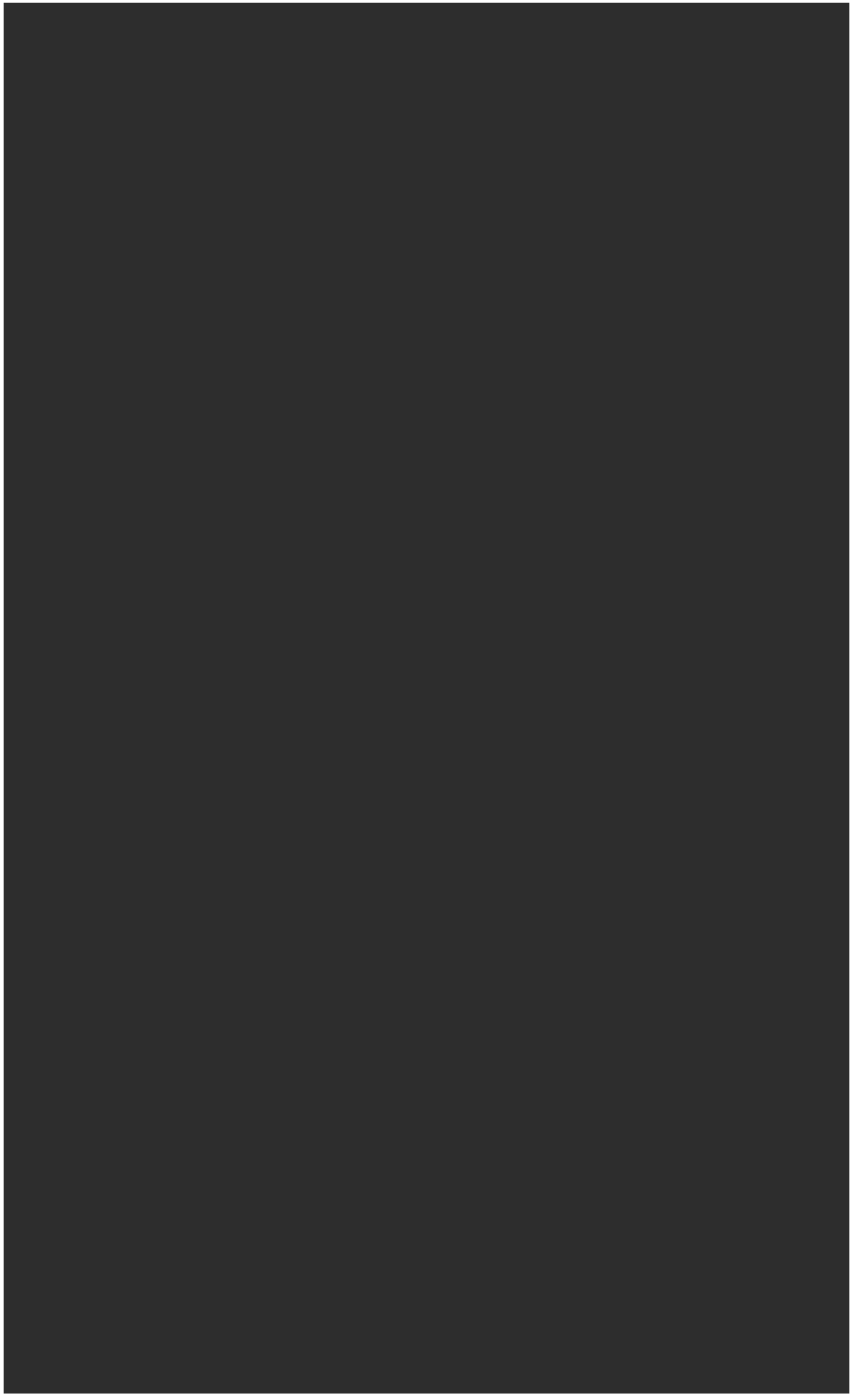
Note

Please note that you may get a report timeout if you select a large time period.

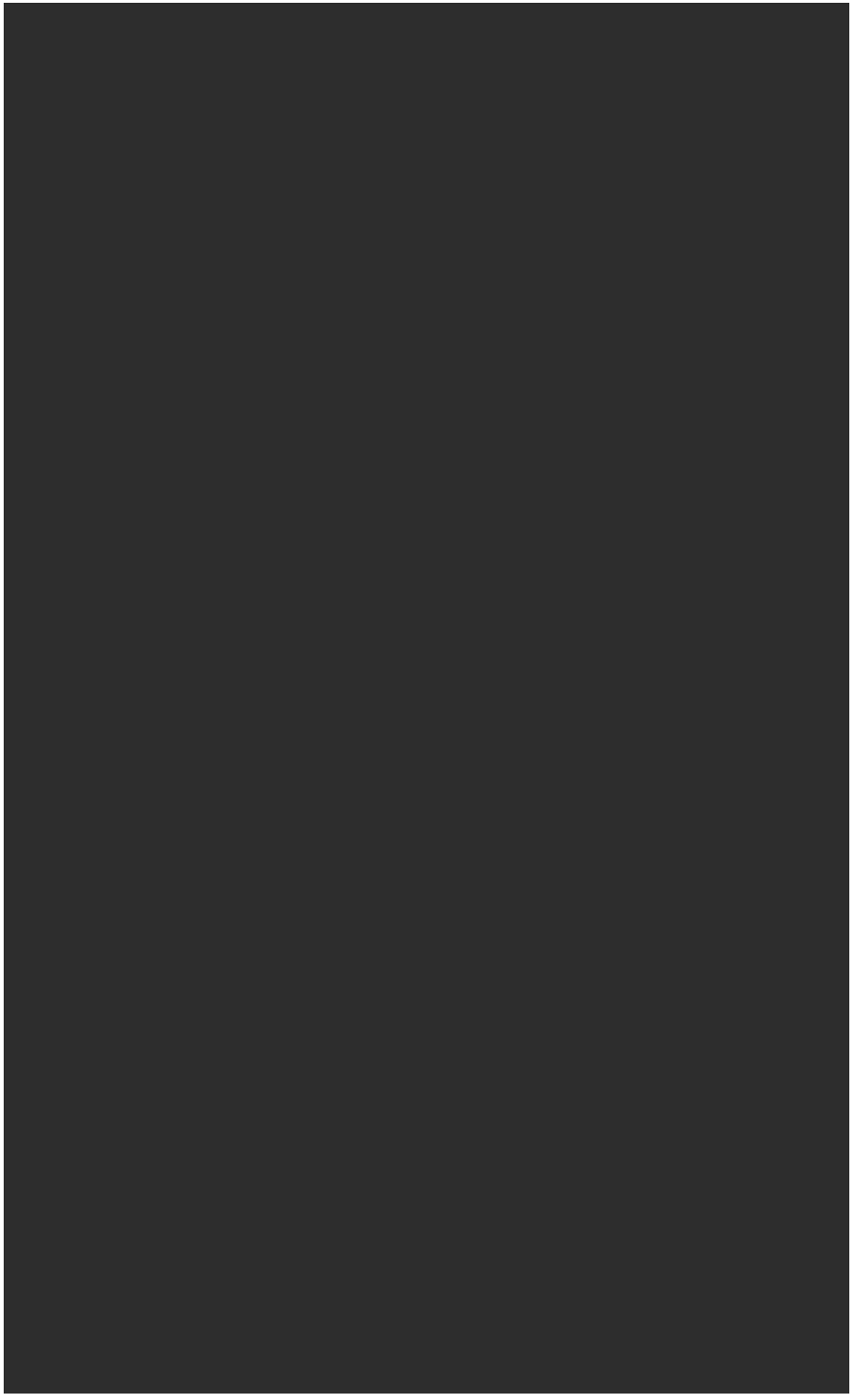
Time reports data source fields

Here are the fields that are aplicable for time reports queries:

Field Name	Description	GDS Type
worked_on	The date when work was performed by the freelancer.	date
week_worked_on	Monday's date corresponding to the week when the *worked_on* occurs.	date
month_worked_on	The month, in number, for the date in the *worked_on* field.	date
year_worked_on	The year in the date that appears in the Start Date field.	date
provider_id	The ID of freelancer.	string
provider_name	The freelancer's name.	string



team_id	The ID of team billed.	string
team_name	The name of team billed.	string
assignment_team_id	The ID of the hiring team in the assignment.	string
assignment_name	The opening title of the assignment.	string
agency_id	The team ID of the agency.	string
agency_name	The name of the agency.	string
company_id	The team ID of rollup `assignment_team_id`.	string
agency_company_id	The agency ID of rollup `agency_id`.	string
task	The activities which the freelancer worked on.	string
memo	The memos logged by the freelancer during work.	string
hours	The total hours worked by the freelancer during the date of `worked_on`. **Supports aggregation**	number
charges	The total amount charged to the client. **Supports aggregation**	number
hours_online	The number of online hours in hours. **Supports aggregation**	number



charges_online	The charges of work performed online. **Supports aggregation**	number
hours_offline	The number of offline hours, in hours. **Supports aggregation**	number
charges_offline	The charges of work performed offline. **Supports aggregation**	number

Note

The format of GDS date type is **[yyyy-mm-dd]**.

Aggregation Functions

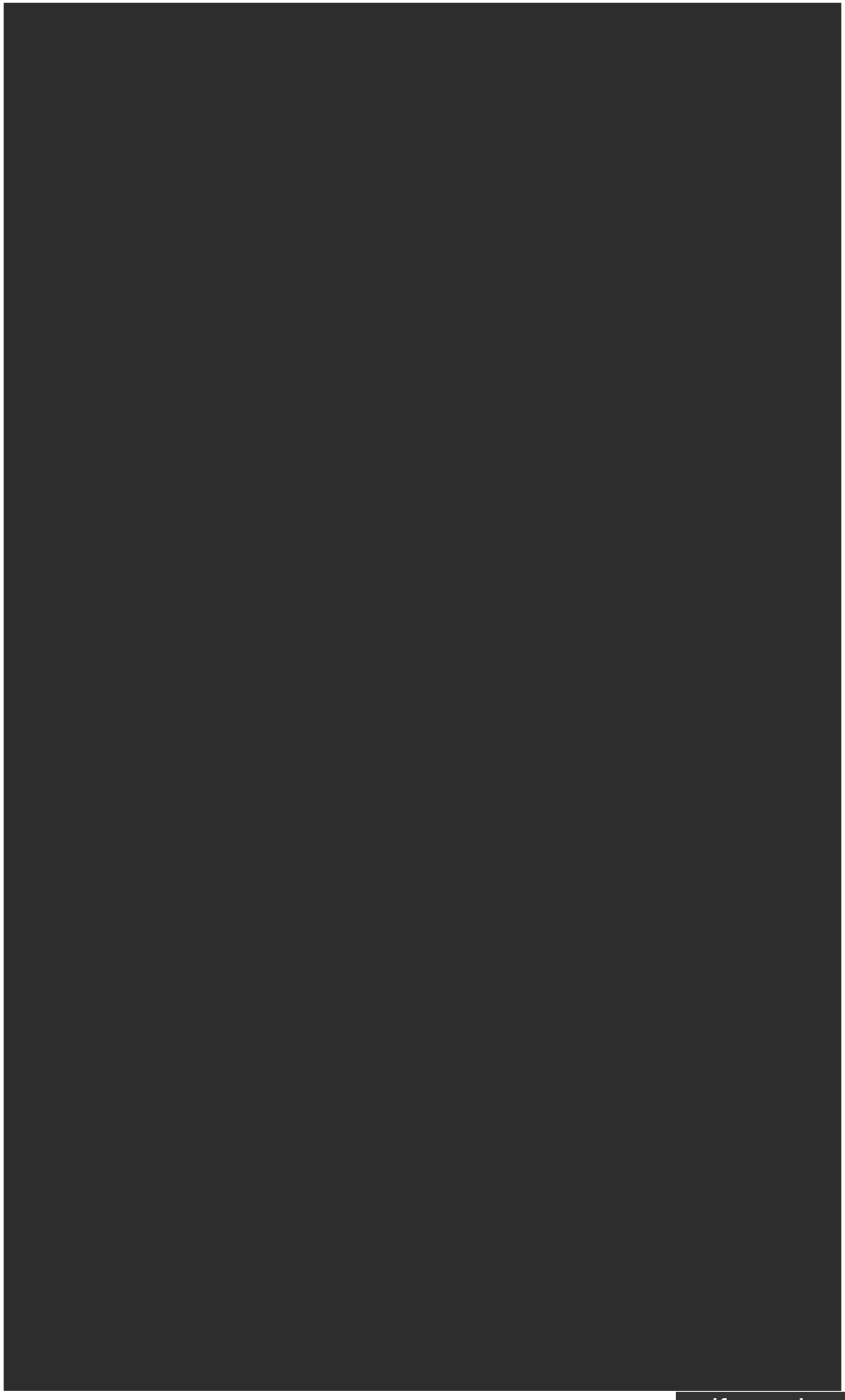
Name	Description	Data Type
sum()	Sums all the values in the columns for a group	Numeric

Note

All fields that co-exist with an aggregated field will be implicitly grouped.

Language Clauses

Clause	Usage
select	Selects which fields from the time reports data source table to return.



where	A condition to match rows.
group by	This clause is not supported but the same effect can be achieved by applying an aggregation function to the columns listed in the **select** clause. For instance, applying `sum()` to one of the columns causes other columns to be grouped implicitly.
order by	Sort rows in the return columns.

Select

Only the fields defined in the data source table can be selected. If this clause is not specified, default fields of specific functions are returned. However, `select \*` is not supported.

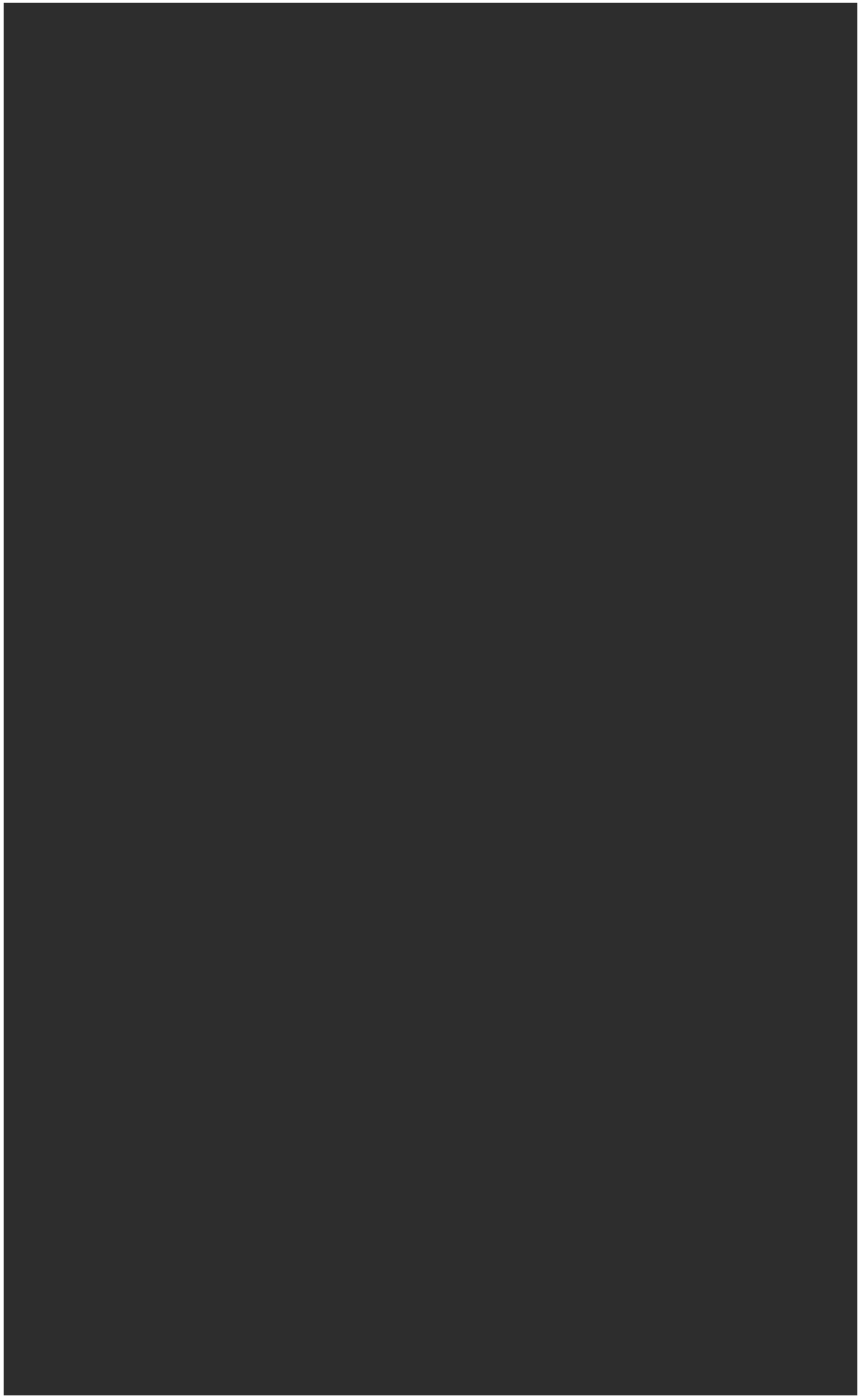
Fields with numeric type specified in the `select` clause can be applied with the Aggregation function, if supported.

Where

The use of a `where` clause is limited to the following fields:

`company_id`, `agency_id`, `provider_id`, `worked_on`, `assignment_team_id`, `task`

Grammar



The grammar of the query is on the left. Multiple conditions can be joined with `and` operator. Multiple conditions matching a specific field can be joined with `or` operator and enclosed in parentheses.

The comparison operators are also limited each field's data types:

Field	Data Type	Operator
company_id	Text	=
agency_id	Text	=
provider_id	Text	=
assignment_team_id	Text	=
task	Text	=
worked_on	Date	> >= = <= <

Order by

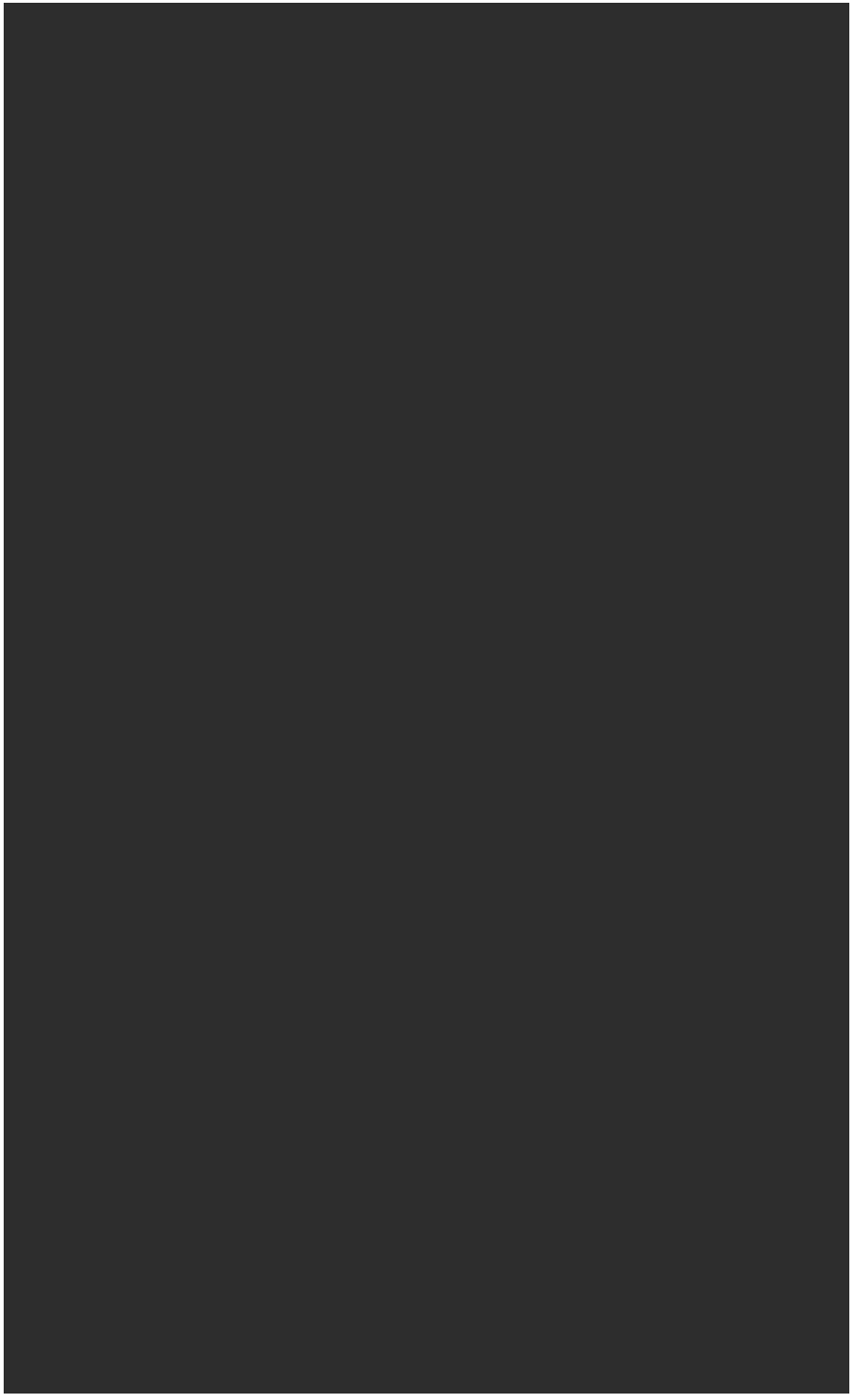
Sorting on row values to all the fields specified in `select` clause is supported.

Financial reports data source fields

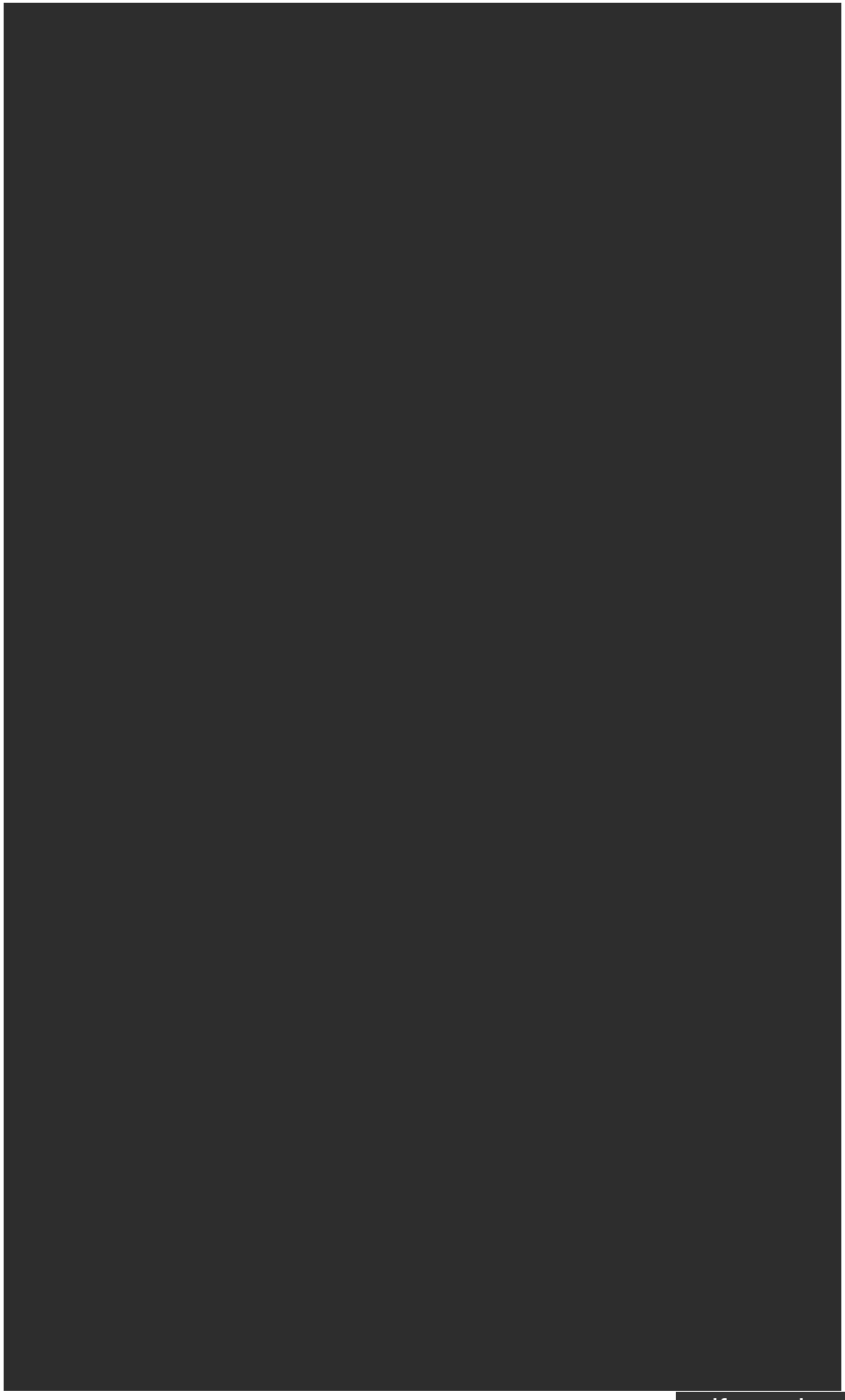
```
WHERE
   (company_id = XYZ) | (agency_id = XYZ)
|
   (provider_id = XYZ)
   and
   [ (worked_on > YYYY-MM-DD) |
(worked_on >= YYYY-MM-DD) |
     (worked_on = YYYY-MM-DD) |
(worked_on <= YYYY-MM-DD) |
     (worked_on < YYYY-MM-DD) ]
   and
   [[ (assignment_team_id = XXX or
assignment_team_id = YYY
       or assignment_team_id = ZZZ) ]
   and
    [ (provider_id = XXX or provider_id =
YYY
       or provider_id = ZZZ) ]]
   and
   [ (task = XXX or task = YYY or task =
ZZZ) ]
```

Fields that are applicable for financial reports queries:

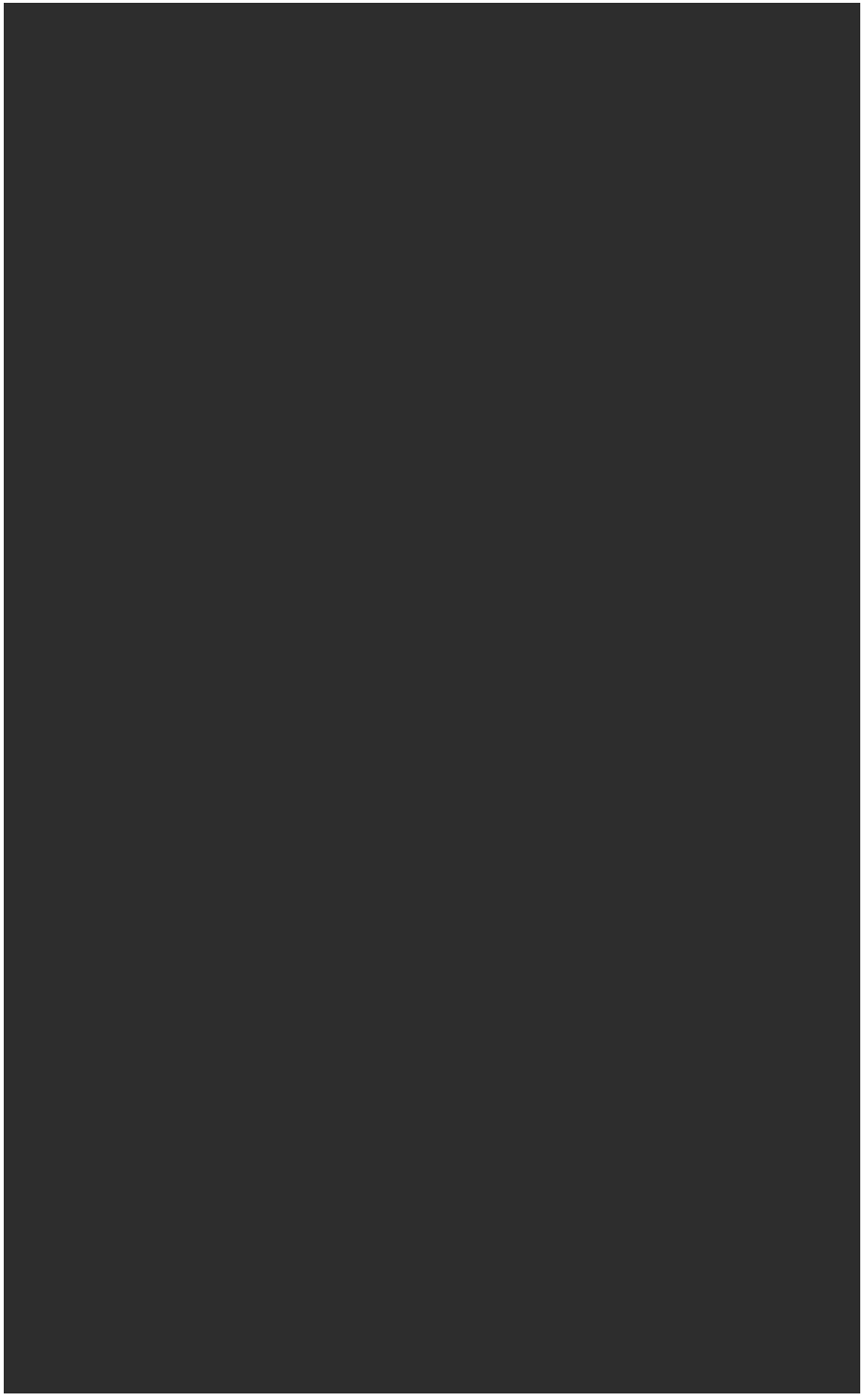
Field Name	Description	GDS Type
reference	The reference of the accounting transaction.	number
date	The date when the transaction was created.	date
date_due	The date when the the accounting transaction was committed.	date
assignment__reference	The reference ID of an assignment for which this payment was made, if available.	number
assignment_name	The name of the assignment.	string
accounting_entity__reference	The reference ID of an `AccountingEntity` used for the accounting transaction.	number



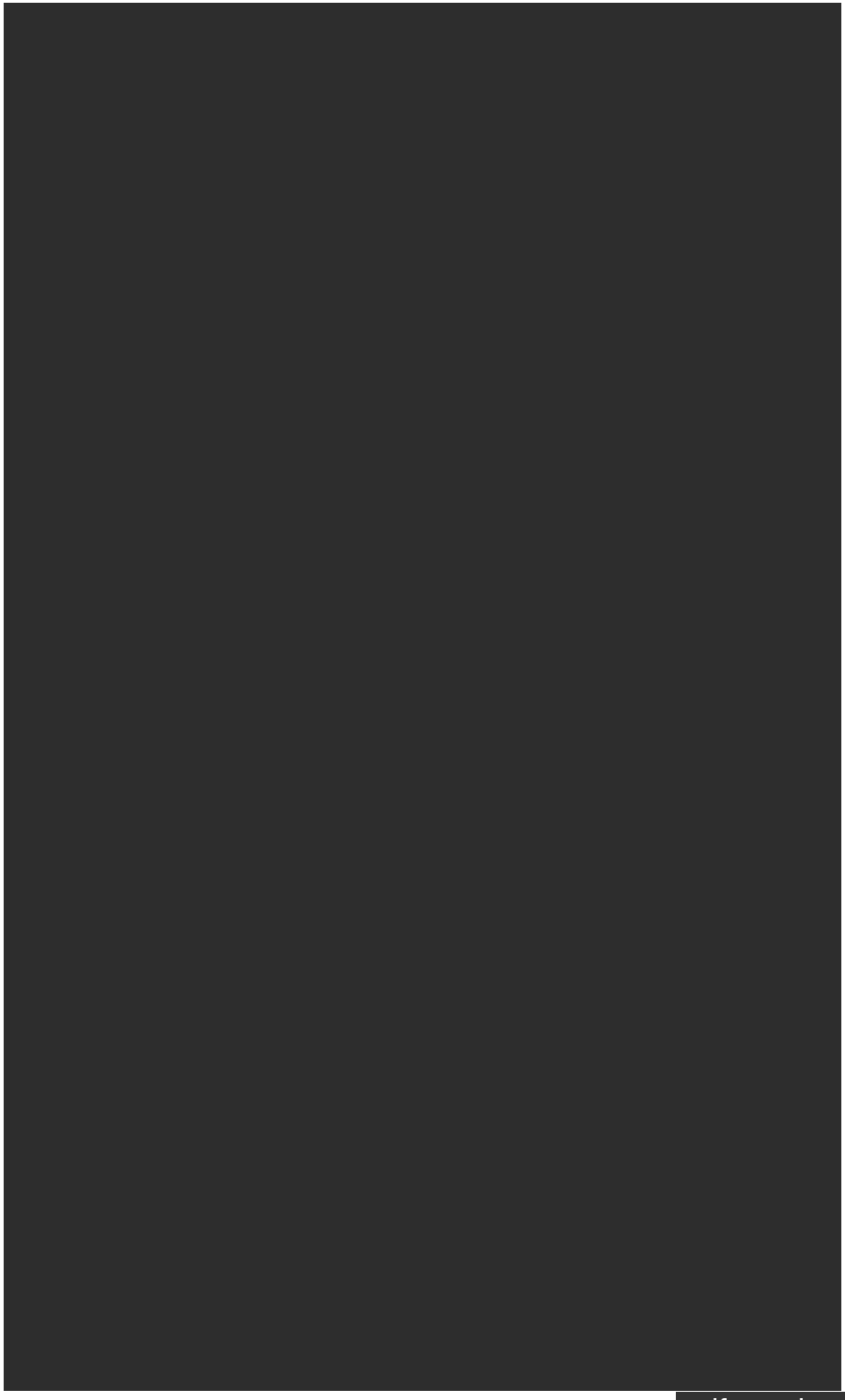
accounting_entity_name	The name of the `AccountingEntity`.	string
buyer_company__reference	The reference ID of the client's company that the client's team belongs to.	number number
buyer_company__id	The literal ID of the client's company.	string
buyer_company_name	The name of the client's company.	string
buyer_team__reference	The reference ID of the client's team that posts owns the assignment.	number
buyer_team__id	The literal ID of the client's team.	string
buyer_team_name	The name of the client's team.	string
provider_company__reference	The reference ID of the freelancer's company.	number
provider_company__id	The literal ID of the freelancer's company.	string
provider_company_name	The name of the freelancer's company.	string



provider_team__reference	The reference ID of the freelancer's team that the freelancer is staffed under for the assignment.	number
provider_team__id	The literal ID of the freelancer's team.	string
provider_team_name	The name of the freelancer's team.	string
provider__reference	The reference ID of the freelancer who works under the assignment.	number
provider__id	The literal ID of the freelancer.	string
provider_name	The name of the freelancer.	string
type	The transaction type - `Invoice`, `Payment`, `Adjustment`.	string
subtype	The detailed transaction type - `Hourly`, `Fixed Price`, `Bonus`, `Refund`, `Withdrawal`,	string



	`Payment`, `Adjustment`, `Salary`, `Security Deposit`, `Chargeback`, `Chargeback Resolution`, `Referral`, `Customer Satisfaction`, `Expense`, `Overtime`, `Milestone`, `Upfront Payment`, `Credit`.	
description	The description of the transaction.	string
comment	The comment given for adjustments.	string
memo	The memo for the transaction.	string
notes	The added by user notes.	string
amount	The amount of the transaction. **Supports aggregation**	number
po_number	The purchase order number.	number



Select

Only the fields defined in the table can be selected. If this clause is not specified or if `select \*` is used, no data is returned.

Fields with numeric type specified in the `select` clause can be applied to the Aggregation function, if supported.

Grammar

The `where` clause is supported on a limited bases as described on the left.

Multiple conditions can be joined by `and` operator. Multiple conditions matching a specific field can be joined by `or` operator and enclosed in parentheses.

The comparison operators are also limited by the supported field's data types:

Field	Data Type	Operator
buyer_company__id	Text	=
buyer_team__id	Text	=
provider_team__id	Text	=
provider__id	Text	=
buyer_company__reference	Numeric	=
buyer_team__reference	Numeric	=

Nested conditions are also supported.

```
WHERE
   [[ (buyer_company__id = 'XYZ') |
(buyer_team__id = 'XYZ') |
      (provider_team__id = 'XYZ') |
(provider__id = 'XYZ')] |
    [ (buyer_company__reference = 1234) |
(buyer_team__reference = 1234) |
     (provider_team__reference = 1234) |
(provider__reference = 1234)] |
     (assignment__reference = 1234) |
(accounting_entity__reference) ]
   and
   [ (date > YYYY-MM-DD) | (date >=
YYYY-MM-DD) |
     (date_due = YYYY-MM-DD) | (date_due
<= YYYY-MM-DD) |
     (date < YYYY-MM-DD) ]
   and
   [[ (buyer_team__id = XXX or
buyer_team__id = YYY
       or buyer_team__id = ZZZ) ]
    and
    [ (provider__id = XXX or provider__id
```

provider_team__reference	Numeric	=
provider__reference	Numeric	=
accounting_entity__reference	Numeric	=
assignment_reference	Numeric	=
type	Text	=
date	Date	> >= = <= <
date_due	Date	> >= = <= <

Order by

Sorting on row values to all the fields specified in `select` clause is supported.

Get time reports for a team (hours only)

Endpoint

GET

/gds/timereports/v1/companies/{company_id}/teams/{team_

```
= YYY
        or provider__id = ZZZ) ]]
   and
   [ (provider_team__id = XXX or
provider_team__id=YYY) ]
   and
   [ (type = 'ARInvoice' or
   (type = 'ARAdjustment' and subtype =
'Fixed Price')) ]
```

DEFINITION

```
reports.getByTeamLimited(
  company, team, params, callback);
```

id}/hours

This call generates time report for a team without detailed monetary information (charges, etc.), only information about hours. Note that if the `provider_id` field is present in the query, the caller must be a supervisor of the freelancers. If the `provider_id` is not provided in the query, the caller must have either hiring manager or finance permission to the specified team.

Required key permissions

Generate time and financial reports for your companies and teams

Arguments

company_id: **required, string**

The company ID. Use Companies & Teams resource to get it.

team_id: **required, string**

The team ID. Use Companies & Teams resource to get it.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Fields that cannot appear in the query: `team_id`, `team_name`, `charges`,

EXAMPLE REQUEST

```
var Time = require('upwork-
api/lib/routers/reports/time.js').Time;

var reports = new Time(api);
var companyId = '123abc';
var teamId = '678def';
var params = {
  'tq': "SELECT worked_on, provider_id,
provider_name, sum(hours) WHERE worked_on
>= '2009-10-01' AND worked_on <= '2009-
10-31'"
};
reports.getByTeamLimited(companyId,
teamId, params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
The structure of the response depends on
the query.

{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405948018,
  'table': {
    # ...
  }
}
```

`charges_online`, `charges_offline`. Example: `tq=SELECT worked_on, provider_id, provider_name, sum(hours) WHERE worked_on >= '2009-10-01' AND worked_on <= '2009-10-31'`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON is returned by default. Example: `tqx=out:xml`.

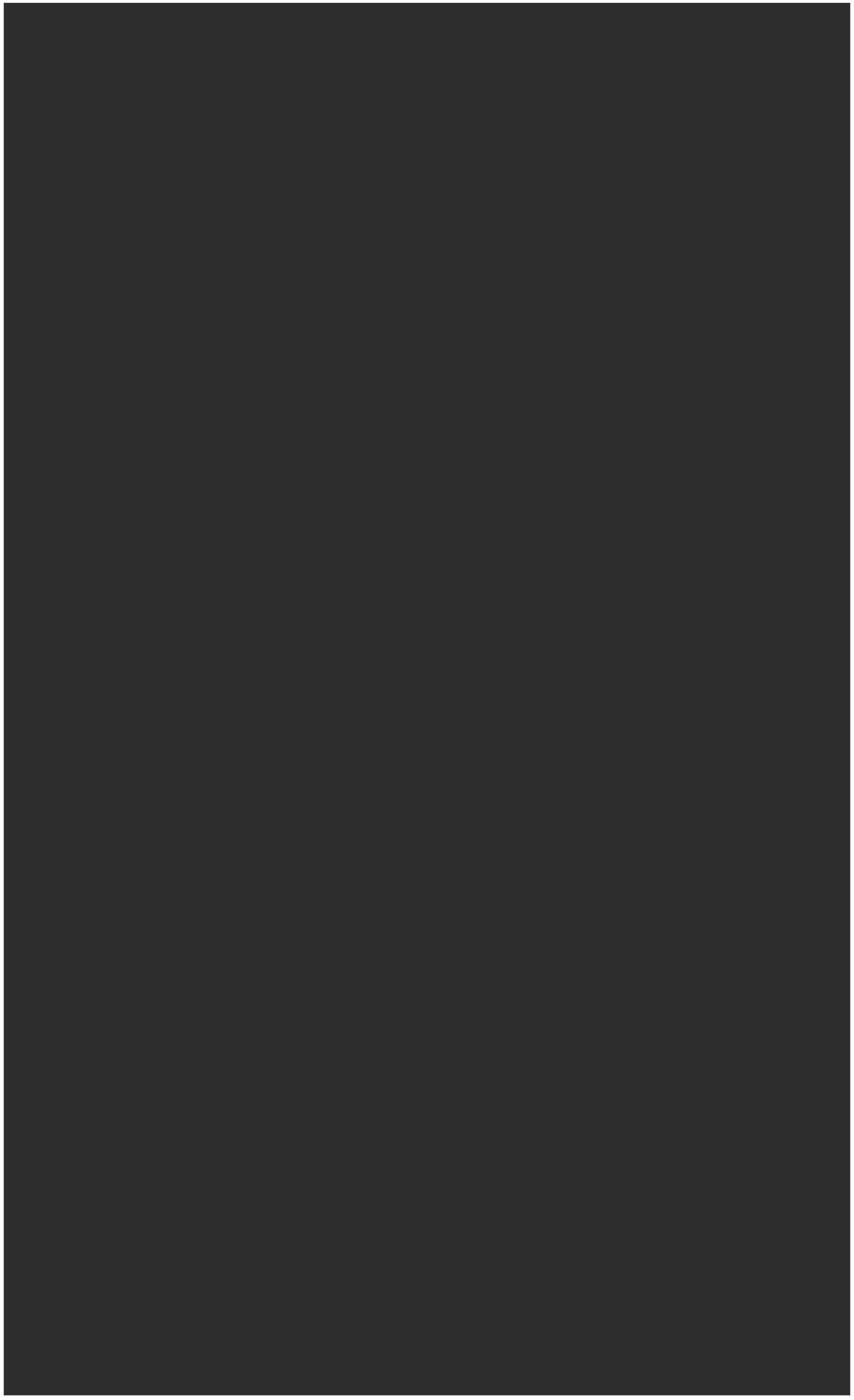
Error messages

401:	Unauthorized
200:	Access denied (user has not enough team/company permissions)
200:	Invalid query
200:	Other (in case of CUSTOM_ERROR exception)

Returns

The data structure returned depends on the query used in the call.

Get time reports for a team



Endpoint

GET

/gds/timereports/v1/companies/{company_id}/teams/{team_id}

This call generates time report for a specific team, with detailed monetary information based on the given query at the time the call is made. Note that if the `provider_id` field is present in the query, the caller must be a supervisor of the freelancers. If the `provider_id` is not provided in the query, the caller must have either hiring manager or finance permission to the specified team.

Required key permissions

Generate time and financial reports for your companies and teams

Arguments

company_id: **required, string**

The company ID. Use Companies & Teams resource to get it.

team_id: **required, string**

The team ID. Use Companies & Teams resource to get it.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google

DEFINITION

```
reports.getByTeamFull(
  company, team, params, callback);
```

EXAMPLE REQUEST

```
var Time = require('upwork-
api/lib/routers/reports/time.js').Time;

var reports = new Time(api);
var companyId = '123abc';
var teamId = '678def';
var params = {
  'tq': "SELECT worked_on, provider_id,
provider_name, sum(hours) WHERE worked_on
>= '2009-10-01' AND worked_on <= '2009-
10-31'"
};
reports.getByTeamFull(companyId, teamId,
params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
The structure of the response depends on
the query.

{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405948018,
  'table': {
    # ...
  }
}
```

Data Source Language. Fields that cannot appear in the query: `team_id`, `team_name`. Example: `tq=SELECT worked_on, provider_id, provider_name, sum(hours) WHERE worked_on >= '2009-10-01' AND worked_on <= '2009-10-31'`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON is returned by default. Example: `tqx=out:xml`.

Error messages

401:	Unauthorized
200:	Access denied (user has not enough team/company permissions)
200:	Invalid query
200:	Other (in case of CUSTOM_ERROR exception)

Returns

The data structure returned depends on the query used in the call.

Get companywide time reports

Endpoint

GET /gds/timereports/v1/companies/{company_id}

Time reports can be generated at company-wide level. In order to access this call the authorized user needs either hiring or finance permissions to all teams within the company.

Required key permissions

Generate time and financial reports for your companies and teams

Arguments

company_id: **required, string**

The company ID. Use Companies & Teams resource to get it.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Fields that cannot appear in the query: `company_id`. Example: `tq=SELECT week_worked_on, assignment_team_id, sum(hours), sum(charges) WHERE worked_on > '2009-10-01' AND worked_on <=

DEFINITION

```
reports.getByCompany(
  company, params, callback);
```

EXAMPLE REQUEST

```
var Time = require('upwork-
api/lib/routers/reports/time.js').Time;

var reports = new Time(api);
var companyId = '123abc';
var params = {
  'tq': "SELECT week_worked_on,
assignment_team_id, provider_id,
assignment_name, sum(hours) WHERE
worked_on > '2009-10-01' AND worked_on <=
'2009-10-31'"
};
reports.getByCompany(companyId, params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
The structure of the response depends on
the query.

{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405948018,
  'table': {
    # ...
```

'2009-10-31'`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON is returned by default. Example: `tqx=out:xml`.

Error messages

401: Unauthorized

200: Access denied (user has not enough team/company permissions)

200: Invalid query

200: Other (in case of CUSTOM_ERROR exception)

Returns

The data structure returned depends on the query used in the call.

Get agency specific time reports

Endpoint

GET

/gds/timereports/v1/companies/{company_id}/agencies/{agency_id}

```
    }
  }
```

DEFINITION

```
report.getByAgency(
  company, agency, params, callback);
```

EXAMPLE REQUEST

Time reports can be generated for an agency. In order to use this call the authorized user needs staffing manager permissions to the agency.

Required key permissions

Generate time and financial reports for your companies and teams

Arguments

company_id: **required, string**

The company ID. Use Companies & Teams resource to get it.

agency_id: **required, string**

The agency ID. Use Companies & Teams resource to get it.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Fields that cannot appear in the query: `agency_id`, `agency_name`, `charges`, `charges_online`, `charges_offline`. Example: `tq=SELECT week_worked_on, assignment_team_id, provider_id, assignment_name, sum(hours) WHERE worked_on > '2009-10-01'

```
var Time = require('upwork-
api/lib/routers/reports/time.js').Time;

var reports = new Time(api);
var companyId = '123abc';
var agencyId = '678def';
var params = {
  'tq': "SELECT week_worked_on,
assignment_team_id, provider_id,
assignment_name, sum(hours) WHERE
worked_on > '2009-10-01' AND worked_on <=
'2009-10-31'"
};
reports.getByAgency(companyId, agencyId,
params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
The structure of the response depends on
the query.

{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405948018,
  'table': {
    # ...
  }
}
```

AND worked_on <= '2009-10-31'`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON is returned by default. Example: `tqx=out:xml`.

Error messages

401:	Unauthorized
200:	Access denied (user has not enough team/company permissions)
200:	Invalid query
200:	Other (in case of CUSTOM_ERROR exception)

Returns

The data structure returned depends on the query used in the call.

Get freelancer specific time reports (hours only)

Endpoint

GET /gds/timereports/v1/providers/{user_id}/hours

This call allows callers to generate time reports for

DEFINITION

```
report.getByFreelancerLimited(
  freelancerId, params, callback);
```

EXAMPLE REQUEST

themselves. No monetary fields, such as charges, are supported. The caller must be the freelancer himself.

Required key permissions

Generate time and financial reports for your companies and teams

Arguments

user_id: **required, string**

The freelancer's user ID. This should be the user ID of the authenticated user.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Fields that cannot appear in the query: `provider_id`, `provider_name`, `charges`, `charges_online`, `charges_offline`. Example: `tq=SELECT worked_on, assignment_team_id, hours, task, memo WHERE worked_on > '2009-10-01' AND worked_on <= '2009-10-31'`.

tqx: **optional, string**

Default: `out:json`

```
var Time = require('upwork-
api/lib/routers/reports/time.js').Time;

var reports = new Time(api);
var freelancerId = 'john_freelancer';
var params = {
  'tq': "SELECT worked_on,
assignment_team_id, hours, task, memo
WHERE worked_on > '2009-10-01' AND
worked_on <= '2009-10-31'"
};
reports.getByFreelancerLimited(freelancer
Id, params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

The structure of the response depends on the query.

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405948018,
  'table': {
    # ...
  }
}
```

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON is returned by default. Example: `tqx=out:xml`.

Error messages

401:	Unauthorized
200:	Access denied (user has not enough team/company permissions)
200:	Invalid query
200:	Other (in case of CUSTOM_ERROR exception)

Returns

The data structure returned depends on the query used in the call.

Get freelancer specific time reports

Endpoint

GET /gds/timereports/v1/providers/{user_id}

This call allows callers to generate time reports for themselves, including monetary information. The caller must be the freelancer himself.

Required key permissions

DEFINITION

```
report.getByFreelancerFull(
  freelancerId, params, callback);
```

EXAMPLE REQUEST

```
var Time = require('upwork-
api/lib/routers/reports/time.js').Time;

var reports = new Time(api);
```

Generate time and financial reports for your companies and teams

Arguments

user_id: **required, string**

The freelancer's user ID. This should be the username of the authenticated user.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Fields that cannot appear in the query: `provider_id`, `provider_name`, `charges`, `charges_online`, `charges_offline`. Example: `tq=SELECT worked_on, assignment_team_id, hours, task, memo WHERE worked_on > '2009-10-01' AND worked_on <= '2009-10-31'`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON

```
var reports = new Time(api);
var freelancerId = 'john_freelancer';
var params = {
  'tq': "SELECT worked_on,
assignment_team_id, hours, task, memo
WHERE worked_on > '2009-10-01' AND
worked_on <= '2009-10-31'"
};
reports.getByFreelancerFull(freelancerId,
params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

The structure of the response depends on the query.

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405948018,
  'table': {
    # ...
  }
}
```

is returned by default. Example: `tqx=out:xml`.

Error messages

401:	Unauthorized
200:	Access denied (user has not enough team/company permissions)
200:	Invalid query
200:	Other (in case of CUSTOM_ERROR exception)

Returns

The data structure returned depends on the query used in the call.

Get billing reports for a freelancer

Endpoint

GET /gds/finreports/*v2*/providers/{provider_ref}/billings

This call allows freelancers to find out what clients are paying for their services. Disallowed fields: `comment`, `po_number`. Supported filters: `date`, `week`, `month`, `year`, `date_due`, `buyer_company__reference`, `buyer_company__id`, `buyer_team__reference`, `buyer_team__id`, `provider_company__reference`, `provider_company__id`, `assignment__reference`, `type`, `subtype`. Permissions: freelancer.

DEFINITION

```
billings.getByFreelancer(
  freelancerReference, params, callback);
```

EXAMPLE REQUEST

```
var Billings = require('upwork-
api/lib/routers/reports/finance/billings.
js').Billings;

var billings = new Billings(api);
var freelancerReference = '1234';
var params = {
  'tq': "SELECT amount WHERE date >=
'2009-10-01' AND date <= '2009-10-31'"
};
```

Required key permissions

Generate time and financial reports for your companies and teams

Arguments

provider_ref: **required, integer**

The reference ID of the freelancer (authenticated user). Use Teams & Companies resource to get it.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Example: `tq=SELECT amount WHERE date >= '2009-10-01' AND date <= '2009-10-31'`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON is returned by default. Example: `tqx=out:xml`.

Error messages

401: Unauthorized

200: User has not enough permissions

```
billings.getByFreelancer(freelancerRefere
nce, params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

The structure of the response depends on the query.

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405948018,
  'table': {
    # ...
  }
}
```

200: Invalid query

200: Other (in case of CUSTOM_ERROR exception)

Returns

The data structure returned depends on the query used in the call.

Get billing reports for a freelancer's team

Endpoint

GET

/gds/finreports/v2/provider_teams/{provider_team_ref}/billings

This call allows the staffing manager of a team to find out what clients are paying for their services. Note, that authenticated user needs to be an admin or staffing manager of the requested team. Disallowed fields: `comment`, `po_number`. Supported filters: `date`, `week`, `month`, `year`, `date_due`, `buyer_company__reference`, `buyer_company__id`, `buyer_team__reference`, `buyer_team__id`, `provider__reference`, `provider__id`, `assignment__reference`, `type`, `subtype`. Permissions: admin or staffing

Required key permissions

DEFINITION

```
billings.getByFreelancersTeam(
  freelancerTeamReference, params,
callback);
```

EXAMPLE REQUEST

```
var Billings = require('upwork-
api/lib/routers/reports/finance/billings.
js').Billings;

var billings = new Billings(api);
var freelancerTeamReference = '1234';
var params = {
  'tq': "SELECT amount WHERE date >=
'2009-10-01' AND date <= '2009-10-31' AND
buyer_team__reference=1234"
};
billings.getByFreelancersTeam(freelancerT
eamReference, params, function(error,
data) {
  console.log(data);
});
```

Generate time and financial reports for your companies and teams

Arguments

provider_team_re **required, integer**

f: The reference ID of the team the authenticated user has access to. The authenticated user must be an admin or a staffing manager of the team. Use Companies & Teams resource to get the reference ID.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Example: `tq=SELECT amount WHERE date >= '2009-10-01' AND date <= '2009-10-31' AND buyer_team__reference=1234`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON is returned by default. Example: `tqx=out:xml`.

EXAMPLE RESPONSE

The structure of the response depends on the query.

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405948018,
  'table': {
    # ...
  }
}
```

Error messages

401:	Unauthorized
200:	User has not enough permissions
200:	Invalid query
200:	Other (in case of CUSTOM_ERROR exception)

Returns

The data structure returned depends on the query used in the call.

Get billing reports for a freelancer's company

Endpoint

GET

/gds/finreports/v2/provider_companies/{provider_company_ref}/billings

This call allows freelancer company owners to know what clients are paying for their AC services, cross multiple accounting entities. Disallowed fields: `comment`, `po_number`. Supported filters: `date`, `week`, `month`, `year`, `date_due`, `buyer_company__reference`, `buyer_company__id`, `buyer_team__reference`, `buyer_team__id`, `provider__reference`, `provider__id`, `assignment__reference`, `accounting_entity__reference`, `type`, `subtype`. Permissions: owner or admin.

DEFINITION

```
billings.getByFreelancersCompany(
  freelancerCompanyReference, params,
  callback);
```

EXAMPLE REQUEST

```
var Billings = require('upwork-
api/lib/routers/reports/finance/billings.
js').Billings;

var billings = new Billings(api);
var freelancerCompanyReference = '1234';
var params = {
  'tq': "SELECT amount WHERE date >=
'2009-10-01' AND date <= '2009-10-31' AND
buyer_company__reference=89089"
};
billings.getByFreelancersCompany(freelanc
erCompanyReference, params,
function(error, data) {
```

Required key permissions

Generate time and financial reports for your companies and teams

Arguments

provider_company_ref: **required, integer**

The reference ID of the company's parent team the authenticated user has access to. The authenticated user must be the owner of the company. Use Teams & Companies resource to get it.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Example: `tq=SELECT amount WHERE date >= '2009-10-01' AND date <= '2009-10-31' AND buyer_company__reference=89089`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON

```
  console.log(data);
});
```

EXAMPLE RESPONSE

```
The structure of the response depends on
the query.

{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405948018,
  'table': {
    # ...
  }
}
```

is returned by default. Example: `tqx=out:xml`.

Error messages

401:	Unauthorized
200:	User has not enough permissions
200:	Invalid query
200:	Other (in case of CUSTOM_ERROR exception)

Returns

The data structure returned depends on the query used in the call.

Get earning reports for a freelancer

Endpoint

GET /gds/finreports/v2/providers/{provider_ref}/earnings

This call allows freelancers to find out what they are being paid for their services. Disallowed fields: `comment`, `po_number`. Supported filters: `date`, `week`, `month`, `year`, `date_due`, `buyer_company__reference`, `buyer_company__id`, `buyer_team__reference`, `buyer_team__id`, `provider_company__reference`, `provider_company__id`, `assignment__reference`, `type`, `subtype`. Permissions: freelancer.

Required key permissions

DEFINITION

```
earnings.getByFreelancer(
  freelancerReference, params, callback);
```

EXAMPLE REQUEST

```
var Earnings = require('upwork-
api/lib/routers/reports/finance/earnings.
js').Earnings;

var earnings = new Earnings(api);
var freelancerReference = '1234';
var params = {
  'tq': "SELECT amount WHERE date >=
'2009-10-01' AND date <= '2009-10-31'"
};
earnings.getByFreelancer(freelancerRefere
nce, params, function(error, data) {
```

Generate time and financial reports for your companies and teams

Arguments

provider_ref: **required, integer**

The reference ID of the freelancer (authenticated user). Use Teams & Companies resource to get it.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Example: `tq=SELECT amount WHERE date >= '2009-10-01' AND date <= '2009-10-31'`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON is returned by default. Example: `tqx=out:xml`.

Error messages

401: Unauthorized

200: User has not enough permissions

200: Invalid query

```
  console.log(data);
});
```

EXAMPLE RESPONSE

The structure of the response depends on the query.

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405948018,
  'table': {
    # ...
  }
}
```

200: Other (in case of CUSTOM_ERROR exception)

Returns

The data structure returned depends on the query used in the call.

Get earning reports for a freelancer's team

Endpoint

GET

/gds/finreports/v2/provider_teams/{provider_team_ref}/earnings

This call allows freelancer's company owners or managers to know what they are being paid per client for their services, and amounts that remain unpaid. Disallowed fields: `comment`, `po_number`. Supported filters: `date`, `week`, `month`, `year`, `date_due`, `buyer_company__reference`, `buyer_company__id`, `buyer_team__reference`, `buyer_team__id`, `provider__reference`, `provider__id`, `assignment__reference`, `type`, `subtype`. Permissions: admin or staffing.

Required key permissions

Generate time and financial reports for your companies and

DEFINITION

```
earnings.getByFreelancersTeam(
  freelancerTeamReference, params,
  callback);
```

EXAMPLE REQUEST

```
var Earnings = require('upwork-
api/lib/routers/reports/finance/earnings.
js').Earnings;

var earnings = new Earnings(api);
var freelancerTeamReference = '1234';
var params = {
  'tq': "SELECT amount WHERE date >=
'2009-10-01' AND date <= '2009-10-31' AND
buyer_team__reference=1234"
};
earnings.getByFreelancerTeam(freelancerTe
amReference, params, function(error,
data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

teams

Arguments

provider_team_re **required, string**

f: The reference ID of the team the authenticated user has access to. The authenticated user must be an admin or a staffing manager of the team. Use Companies & Teams resource to get the reference ID.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Example: `tq=SELECT amount WHERE date >= '2009-10-01' AND date <= '2009-10-31' AND buyer_team__reference=1234`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON is returned by default. Example: `tqx=out:xml`.

Error messages

```
The structure of the response depends on
the query.

{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405948018,
  'table': {
    # ...
  }
}
```

401:	Unauthorized
200:	User has not enough permissions
200:	Invalid query
200:	Other (in case of CUSTOM_ERROR exception)

Returns

The data structure returned depends on the query used in the call.

Get earning reports for a freelancer's company

Endpoint

GET

/gds/finreports/v2/provider_companies/{provider_company_ref}/earnings

This call allows freelancer's company owners to know what they are being paid per client for their services, and amounts that remain unpaid, cross multiple accounting entities. Disallowed fields: `comment`, `po_number`. Supported filters: `date`, `week`, `month`, `year`, `date_due`, `buyer_company__reference`, `buyer_company__id`, `buyer_team__reference`, `buyer_team__id`, `provider__reference`, `provider__id`, `assignment__reference`, `accounting_entity__reference`, `type`, `subtype`. Permissions: admin or staffing.

DEFINITION

```
earnings.getByFreelancersCompany(
  freelancerCompanyReference, params,
  callback);
```

EXAMPLE REQUEST

```
var Earnings = require('upwork-
api/lib/routers/reports/finance/earnings.
js').Earnings;

var earnings = new Earnings(api);
var freelancerCompanyReference = '1234';
var params = {
  'tq': "SELECT amount WHERE date >=
'2009-10-01' AND date <= '2009-10-31' AND
buyer_company__reference=89089"
};
earnings.getByFreelancerCompany(freelance
rCompanyReference, params,
function(error, data) {
  console.log(data);
});
```

Required key permissions

Generate time and financial reports for your companies and teams

Arguments

provider_company_ref: **required, string**

The reference ID of the company's parent team the authenticated user has access to. The authenticated user must be the owner of the company. Use Teams & Companies resource to get it.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Example: `tq=SELECT amount WHERE date >= '2009-10-01' AND date <= '2009-10-31' AND buyer_company__reference=89089`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON

EXAMPLE RESPONSE

The structure of the response depends on the query.

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405948018,
  'table': {
    # ...
  }
}
```

is returned by default. Example: `tqx=out:xml`.

Error messages

401:	Unauthorized
200:	User has not enough permissions
200:	Invalid query
200:	Other (in case of CUSTOM_ERROR exception)

Returns

The data structure returned depends on the query used in the call.

Get billing reports for a client's team

Endpoint

GET

/gds/finreports/v2/buyer_teams/{buyer_team_ref}/billings

This call allows admins or hiring managers of a team to see what their freelancers get paid. Disallowed fields: `comment`, `po_number`. Supported filters: `date`, `week`, `month`, `year`, `date_due`, `provider_team__reference`, `provider_team__id`, `provider__reference`, `provider__id`, `assignment__reference`, `type`, `subtype`. Permissions: admin or hiring.

Required key permissions

DEFINITION

```
billings.getByBuyersTeam(
  buyerTeamReference, params, callback);
```

EXAMPLE REQUEST

```
var Billings = require('upwork-
api/lib/routers/reports/finance/billings.
js').Billings;

var billings = new Billings(api);
var buyerTeamReference = '1234';
var params = {
  'tq': "SELECT amount WHERE date >=
'2009-10-01' AND date <= '2009-10-31' AND
provider__reference = '23423'"
};
billings.getByBuyersTeam(buyerTeamReferen
```

Generate time and financial reports for your companies and teams

Arguments

buyer_team_ref: **required, integer**

The reference ID of the team the authenticated user has access to. The authenticated user must be an admin or a staffing manager of the team. Use Companies & Teams resource to get it.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Example: `tq=SELECT amount WHERE date >= '2009-10-01' AND date <= '2009-10-31' AND provider__reference = '23423'`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON is returned by default. Example: `tqx=out:xml`.

```
billings.getByBuyerTeam(buyerTeamReferen
ce, params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

The structure of the response depends on the query.

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405948018,
  'table': {
    # ...
  }
}
```

Error messages

401:	Unauthorized
200:	User has not enough permissions
200:	Invalid query
200:	Other (in case of CUSTOM_ERROR exception)

Returns

The data structure returned depends on the query used in the call.

Get billing reports for a client’s company

Endpoint

GET

/gds/finreports/v2/buyer_companies/{buyer_company_ref}/billings

This call allows company owners to see what their freelancers get paid. Disallowed fields: none. Supported filters: `date`, `week`, `month`, `year`, `date_due`, `provider_company__reference`, `provider_company__id`, `provider__reference`, `provider__id`, `assignment__reference`, `type`, `subtype`. Permissions: owner or admin.

Required key permissions

Generate time and financial reports for your companies and

DEFINITION

```
billings.getByBuyersCompany(
  buyerCompanyReference, params,
  callback);
```

EXAMPLE REQUEST

```
var Billings = require('upwork-
api/lib/routers/reports/finance/billings.
js').Billings;

var billings = new Billings(api);
var buyerCompanyReference = '1234';
var params = {
  'tq': "SELECT amount WHERE date >=
'2009-10-01' AND date <= '2009-10-31'"
};
billings.getByBuyersCompany(buyerCompanyR
eference, params, function(error, data) {
  console.log(data);
});
```

teams

Arguments

buyer_company_ref: **required, string**

The reference ID of the company the authenticated user has access to. The authenticated user must be the owner of the company. Use Companies & Teams resource to get it.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Example: `tq=SELECT amount WHERE date >= '2009-10-01' AND date <= '2009-10-31'`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON is returned by default. Example: `tqx=out:xml`.

Error messages

401: Unauthorized

EXAMPLE RESPONSE

The structure of the response depends on the query.

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405948018,
  'table': {
    # ...
  }
}
```

200: User has not enough permissions

200: Invalid query

200: Other (in case of CUSTOM_ERROR exception)

Returns

The data structure returned depends on the query used in the call.

Get earning reports for a client's team

Endpoint

GET

/gds/finreports/v2/buyer_teams/{buyer_team_ref}/earnings

This call allows hiring managers of a team to see what they pay to their freelancers. Disallowed fields: none. Supported filters: `date`, `week`, `month`, `year`, `date_due`, `provider_team__reference`, `provider_team__id`, `provider__reference`, `provider__id`, `assignment__reference`, `type`, `subtype`, `po_number`. Permissions: hiring.

Required key permissions

Generate time and financial reports for your companies and teams

Arguments

buyer_team_ref: **required, string**

DEFINITION

```
earnings.getByBuyersTeam(
  buyerTeamReference, params, callback);
```

EXAMPLE REQUEST

```
var Earnings = require('upwork-
api/lib/routers/reports/finance/earnings.
js').Earnings;

var earnings = new Earnings(api);
var buyerTeamReference = '1234';
var params = {
  'tq': "SELECT amount WHERE date >=
'2009-10-01' AND date <= '2009-10-31' AND
provider__id = 'jsmith'"
};
earnings.getByBuyersTeam(buyerTeamReferen
ce, params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

The reference ID of the team the authenticated user has access to. The authenticated user must be an admin or a staffing manager of the team. Use Companies & Teams resource to get it.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Example: `tq=SELECT amount WHERE date >= '2009-10-01' AND date <= '2009-10-31' AND provider__id = 'jsmith'`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON is returned by default. Example: `tqx=out:xml`.

Error messages

401:	Unauthorized
200:	User has not enough permissions
200:	Invalid query
200:	Other (in case of CUSTOM_ERROR

```
The structure of the response depends on
the query.

{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405948018,
  'table': {
    # ...
  }
}
```

exception)

Returns

The data structure returned depends on the query used in the call.

Get earning reports for a client's company

Endpoint

GET

/gds/finreports/v2/buyer_companies/{buyer_company_ref}/earnings

This call allows company owners to see what they pay to their freelancers. Disallowed fields: none. Supported filters: `date`, `week`, `month`, `year`, `date_due`, `provider_company__reference`, `provider_company__id`, `provider__reference`, `provider__id`, `assignment__reference`, `type`, `subtype`, `po_number`. Permissions: owner or admin.

Required key permissions

Generate time and financial reports for your companies and teams

Arguments

buyer_company_ref:	**required, string**	The reference ID of the company the

DEFINITION

```
earnings.getByBuyersCompany(
  buyerCompanyReference, params,
  callback);
```

EXAMPLE REQUEST

```
var Earnings = require('upwork-
api/lib/routers/reports/finance/earnings.
js').Earnings;

var earnings = new Earnings(api);
var buyerCompanyReference = '1234';
var params = {
  'tq': "SELECT amount WHERE date >=
'2009-10-01' AND date <= '2009-10-31' AND
provider__id = 'jsmith'"
};
earnings.getByBuyersCompany(buyerCompanyR
eference, params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
The structure of the response depends on
the query.
```

authenticated user has access to. The authenticated user must be the owner of the company. Use Companies & Teams resource to get it.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Example: `tq=SELECT amount WHERE date >= '2009-10-01' AND date <= '2009-10-31' AND provider__id = 'jsmith'`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON is returned by default. Example: `tqx=out:xml`.

Error messages

401:	Unauthorized
200:	User has not enough permissions
200:	Invalid query
200:	Other (in case of CUSTOM_ERROR exception)

```
{
  'auth_user': {
    'first_name': 'John',
    'last_name': 'Johnson',
    'timezone': 'Asia/Omsk',
    'timezone_offset': '25200'
  },
  'server_time': 1405948018,
  'table': {
    # ...
  }
}
```

Returns

The data structure returned depends on the query used in the call.

Get financial reports for an account

Endpoint

GET

/gds/finreports/v2/financial_accounts/{accounting_entity_ref}

This call allows a user to view his/her own accounting transactions or all transactions with accounting entities tied to teams to which the user has hiring manager or finance permissions. Disallowed fields: none. Supported filters: `date`, `week`, `month`, `year`, `date_due`, `provider_company__reference`, `provider_company__id`, `provider__reference`, `provider__id`, `buyer_company__reference`, `buyer_company__id`, `buyer_team__reference`, `buyer_team__id`, `assignment__reference`, `type`, `subtype`, `po_number`, `provider_team__reference`, `provider_team__id`. Permissions: finance manager.

Required key permissions

Generate time and financial reports for your companies and teams

Arguments

DEFINITION

```
finreports.getSpecific(
  entityReference, params, callback);
```

EXAMPLE REQUEST

```
var Accounts = require('upwork-
api/lib/routers/reports/finance/accounts.
js').Accounts;

var finreports = new Accounts(api);
var entityReference = '1234';
var params = {
  'tq': "SELECT amount WHERE date >=
'2009-10-01' AND date <= '2009-10-31'"
};
finreports.getSpecific(entityReference,
params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
The structure of the response depends on
the query.

{
  'auth_user': {
    'first_name': 'John',
```

Arguments

accounting_entity_ref: **required, integer**

The reference ID of an accounting entity, should be a valid user reference ID. Example: `34567`. You can get it with Companies & Teams resource.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Example: `tq=SELECT amount WHERE date >= '2009-10-01' AND date <= '2009-10-31'`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON is returned by default. Example: `tqx=out:xml`.

Error messages

401:	Unauthorized
200:	User has not enough permissions
200:	Invalid query
200:	Other (in case of CUSTOM_ERROR

```
        'last_name': 'Johnson',
        'timezone': 'Asia/Omsk',
        'timezone_offset': '25200'
      },
      'server_time': 1405948018,
      'table': {
        # ...
      }
    }
```

exception)

Returns

The data structure returned depends on the query used in the call.

Get financial reports for the current user's accounts

Endpoint

GET

/gds/finreports/v2/financial_account_owner/{provider_ref}

This call allows a user to view all transactions using accounting entities owned by the user. Disallowed fields: none. Supported filters: `date`, `week`, `month`, `year`, `date_due`, `provider_company__reference`, `provider_company__id`, `provider__reference`, `provider__id`, `buyer_team__reference`, `buyer_team__id`, `assignment__reference`, `accounting_entity__reference`, `type`, `subtype`, `po_number`. Permissions: freelancer.

Required key permissions

Generate time and financial reports for your companies and teams

Arguments

provider_ref: **required, integer**

The reference ID of the freelancer

DEFINITION

```
finreports.getOwned(
  freelancerReference, params, callback);
```

EXAMPLE REQUEST

```
var Accounts = require('upwork-
api/lib/routers/reports/finance/accounts.
js').Accounts;

var finreports = new Accounts(api);
var freelancerReference = '1234';
var params = {
  'tq': "SELECT amount, notes WHERE date
>= '2009-10-01' AND date <= '2009-10-31'"
};
finreports.getOwned(freelancerReference,
params, function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
The structure of the response depends on
the query.

{
  'auth_user': {
```

(authenticated user). Use Teams & Companies resource to get it.

Parameters

tq: **required, string**

The Google query. This should be the valid expression in Upwork Google Data Source Language. Example: `tq=SELECT amount, notes WHERE date >= '2009-10-01' AND date <= '2009-10-31' and (accounting_entity__reference = '1111' OR accounting_entity__reference = '2222')`.

tqx: **optional, string**

Default: `out:json`

Valid values: `out:xml, out:json, out:csv`

The return format. It can be in either XML, JSON or CSV formatting. JSON is returned by default. Example: `tqx=out:xml`.

Error messages

401:	Unauthorized
200:	User has not enough permissions
200:	Invalid query
200:	Other (in case of CUSTOM_ERROR

```
        'first_name': 'John',
        'last_name': 'Johnson',
        'timezone': 'Asia/Omsk',
        'timezone_offset': '25200'
      },
      'server_time': 1405948018,
      'table': {
        # ...
      }
    }
```

exception)

Returns

The data structure returned depends on the query used in the call.

Metadata

This section describes resources that list different “static” data used in Upwork profiles. These resources help you avoid hardcoded data in your applications and keep the apps up to date with Upwork’s future changes.

List categories V2

Endpoint

GET /api/profiles/v2/metadata/categories.{format}

Returns a list of categories for a job/freelancer profile from new structure.

Required key permissions

No special permissions are required

Arguments

format: **required, string**

Default: `json`

Valid values: `json`

DEFINITION

```
metadata.getCategoriesV2(callback);
```

EXAMPLE REQUEST

```
var Metadata = require('upwork-
api/lib/routers/metadata.js').Metadata;

var metadata = new Metadata(api);
metadata.getCategoriesV2(function(error,
data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

Response format.

Error messages

401:	Unauthorized
400:	Bad request

Returns

This call returns the ID and title of the topics within the Category 2.0.

```
[
  {
    'title': 'Web & Mobile Development',
    'id': '531770282580668418'
    'topics': [
      {'id': '531770282589057025',
'title': 'Desktop Software Development'},
      {'id': '531770282589057026',
'title': 'Ecommerce Development'},
       # Another topic
      {'id': '531770282589057032',
'title': 'Other - Web & Mobile
Development'}
    ]
  },
  {
    # Another category
  },
  # ...
]
```

List skills

Endpoint

GET /api/profiles/v1/metadata/skills.{format}

Returns a list of skills available in a freelancer's profile.

Required key permissions

No special permissions are required

Arguments

format:	**optional, string**
	Default: `xml`

DEFINITION

```
metadata.getSkills(callback);
```

EXAMPLE REQUEST

```
var Metadata = require('upwork-
api/lib/routers/metadata.js').Metadata;

var metadata = new Metadata(api);
metadata.getSkills(function(error, data)
{
  console.log(data);
});
```

Valid values: `json, xml`

Response format.

Error messages

401: Unauthorized

400: Bad request

Possible output fields

(Click here to show/hide)

Returns

This call returns the list of skills available in the freelancer's profile.

EXAMPLE RESPONSE

```
[
  '1shoppingcart',
  '2d-animation',
  '2d-design',
  '3d-animation',
  '3d-design',
  '3d-modeling',
  '3d-printing',
  '3d-rendering',
  '3d-rigging',
  # ...
]
```

List geographical regions

Endpoint

GET /api/profiles/v1/metadata/regions.{format}

Returns a list of county regions.

Required key permissions

No special permissions are required

Arguments

format: **optional, string**

Default: `xml`

Valid values: `json, xml`

Response format.

Error messages

DEFINITION

```
metadata.getRegions(callback);
```

EXAMPLE REQUEST

```
var Metadata = require('upwork-
api/lib/routers/metadata.js').Metadata;

var metadata = new Metadata(api);
metadata.getRegions(function(error, data)
{
  console.log(data);
});
```

EXAMPLE RESPONSE

```
[
  {'alias': 'australasia', 'title':
```

401: Unauthorized

400: Bad request

Returns

This call returns a list with title and alias for geographical regions.

List tests

Endpoint

GET /api/profiles/v1/metadata/tests.{format}

Returns a list of available tests at Upwork.

Required key permissions

No special permissions are required

Arguments

format: optional, string

Default: `xml`

Valid values: `json, xml`

Response format.

Error messages

```
  {'alias': 'australasia', 'title':
'Australasia'},
  {'alias': 'east-asia', 'title': 'East
Asia'},
  {'alias': 'eastern-europe', 'title':
'Eastern Europe'},
  {'alias': 'latin-america', 'title':
'Latin America'},
  {'alias': 'misc', 'title': 'Misc'},
  {'alias': 'north-america', 'title':
'North America'},
  {'alias': 'south-asia', 'title': 'South
Asia'},
  {'alias': 'western-europe', 'title':
'Western Europe'}
]
```

DEFINITION

```
metadata.getTests(callback);
```

EXAMPLE REQUEST

```
var Metadata = require('upwork-
api/lib/routers/metadata.js').Metadata;

var metadata = new Metadata(api);
metadata.getTests(function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

```
[
  {
```

401: Unauthorized

400: Bad request

Returns

This call returns a list with the name, weight and record ID of the tests available in the freelancer's profile.

```
    'name': 'Google Android Programming',
    'record_id': '271',
    'weight': '224'
  },
  {
    'name': 'U.S. English AP Style
Editing Skills Test (For Writing
Professionals)',
    'record_id': '374',
    'weight': '182'
  },
  {
    # Another skill
  },
  # ...
]
```

List reasons

Endpoint

GET /api/profiles/v1/metadata/reasons.{format}

Returns a list of reasons by specified type.

Required key permissions

No special permissions are required

Arguments

format: optional, string

Default: `xml`

Valid values: `json, xml`

Response format.

Parameters

DEFINITION

```
metadata.getReasons(params, callback);
```

EXAMPLE REQUEST

```
var Metadata = require('upwork-
api/lib/routers/metadata.js').Metadata;

var metadata = new Metadata(api);
var params = {'type': 'CloseOpening'};
metadata.getReasons(params,
function(error, data) {
  console.log(data);
});
```

EXAMPLE RESPONSE

type:	**required, string** **Valid values:** `EmployerEndsNoStartContract, CloseOpening, RejectCandidate, RejectInterviewInvite, CancelCandidacy, EndProviderContract, EndCustomerContract, EndAssignment` The reason type.

Error messages

401:	Unauthorized
400:	Bad request

Returns

This call returns a list with reason reference number, name and alias.

```
[
  [
    '67',
    'Accidental job posting creation',
    None
  ],
  [
    '51',
    'All positions filled',
    'API_REAS_ALL_POSITIONS_FILLED'
  ],
  [
    '49',
    'Filled by alternate source',
    None
  ],
  [
    '34',
    'No freelancer for requested skills',
    None
  ],
  [
    '41',
    'Project was cancelled',
    None
  ]
]
```