REST API Specification

(OneShift v1)

Version	Date	Author	Description
1.0.0	15-Jul-2014	Dale Fraser	Initial Version
1.0.1	28-Jul-2014	Dale Fraser	Minor improvements
1.0.2	10-Sep-2014	Dale Fraser	Minor Edits
1.0.3	22-Apr-2015	Walter Palombini	Updates
1.0.4	05-May-2015	Dale Fraser	Add categoryId to profession fetch

Summary

Overview

This document describes the OneShift REST based API version 1

The URL required to invoke the rest based API is
https://api.oneshiftjobs.com/
and
https://sandbox.oneshiftjobs.com/

NOTE: Your account must support API authentication before either api or sandbox will work, you need to request API access from OneShift.

Basic Authentication

Every request requires basic authentication. Basic authentication requires a base64 encoded header representing username:password. Basic authentication requires all requests are undertaken over SSL and thus all API requests require you use https.

Request

Type	Params	Values
`HEAD`	`Authorization:`	`Basic Username:Password`

Important

The Username:Password must be a base64 encoded string of the combined username:password, including the colon. The Basic[SPACE] part is NOT base64 encoded.

Example

An example Authorization header parameter would look like this for test@oneshift.com.au:myPass1

Authorization: Basic dGVzdEBvbmVzaGlmdC5jb20uYXU6bXlQYXNzMQ==

Document Conventions

(optional)	dictates optional fields
[value]	dictates the default value of optional fields
*readonly	field is only readable

Results

Format

All actions will generate a Response code of 200 OK where the request was successful. This happens regardless of the data returned. Thus if you attempt to GET an invalid ID or DELETE an invalid ID as long as the request is valid the resulting code is 200 OK.

To determine what is returned you can traverse the resulting data. Every result returns JSON and every return has the following items

Value	URL
`success`	This is a Boolean result true / false which determines if the request returned what you are expecting. Thus requesting an invalid id will result in a status code of 200 OK but with a success result of false.
`message`	This is a human readable message which attempts to identify to the developer a summary of what action was taken. It will be helpful to look at the success and message when developing against the API.

Definition

Job Fields

Details of the fields for the job api

Name	Params	Values
jobId	OneShift ID of job	Required for put
title	Job title	
description	Job description	
type	Job type	casual, permanent
locationId	Location Id	
vehicle	Require Vehicle	(optional) [0]
educationId	Education Level	(optional) [1]
distance	Distance from job km	(optional) [50]
createdAt	Date & Time Created UTC	*readonly
closedAt	Date & Time Closed UTC	*readonly
activationDays	Number of days job is open	*readonly [15]
salaryLowerLimit	Lower Salary Range	(optional) [0]
salaryUpperLimit	Upper Salary Range	(optional) [1000]
salaryType	Salary type	(optional) [hour] / annum
salaryDisplay	Display the salary	(optional) [0]
professionId	Profession Id	
reference	Your unique reference #	
applicationUrl	Url to send application to	Url or Email if you require applications in your system.
applicationEmail	Email to send application to	

Job Arrays

Details of the array fields for the job api

Name	Params	Values
`skills`	`Array of skill ids`	`(optional)`
`certificates`	`Array of certificate ids`	`(optional)`
`languages`	`Array of language ids`	`(optional) [[1]]`

Methods

Job GET

Return a list of jobs for the given authenticated account

Request

Method	URL
GET	/api/v1/job/:id

Parameters

Type	Params	Values
HEAD	Authorization:	Basic Username:Password
HEAD	Content-Type:	application/json
URL	:id	JobId (optional)
URLPARAM	closed=:closed	true/false (optional) [false]
URLPARAM	reference=:reference	(optional)

Examples

Method	URL	Result
GET	/api/v1/job	Return all open jobs
GET	/api/v1/job?closed=true	Return all jobs including closed
GET	/api/v1/job/123	Return job 123
GET	/api/v1/job/123?closed=true	Return job 123 even if its closed
GET	/api/v1/job?reference=ab123	Return job matching reference

Response

<table>
<tr><th>Status</th><th>Response</th></tr>
<tr><td>200</td><td>

```
{
    "success": true,
    "message": "Jobs found.",
    "Jobs": [
        {
            "businessId": 21224,
            "jobId": 40619,
            "jobTitle": "Front End Web Developer",
            "createdAt": "2014-07-15T23:36:55.66",
            "closedAt": null,
            "salaryDisplay": false,
            "salaryLowerLimit": 60000,
            "salaryUpperLimit": 80000,
            "salaryType": "annum",
            "description": "<p>We are seeking the services of a
top notch front end web developer.</p>
<p> You will be working on a high profile website. You must have
a lot of experience in the following</p>
<ul>
<li>CSS</li>
<li>Bootstrap</li>
</ul>
<p>If you are interested in a great company culture in a very
agile environment, please apply</p>",
            "professionId": 161,
            "locationId": 494,
            "educationId": 1,
            "skills": [
                725,
                899
            ],
            "languages": [
                1
            ],
            "certificates": [ ],
            "jobType": "permanent",
            "distance": 50,
            "vehicle": false,
            "activationDays": null
        }
    ]
}
```

</td></tr>
<tr><td>200</td><td>

```
{
    "success": false,
    "message": "No jobs found.",
    "Jobs": [ ]
}
```

</td></tr>
</table>

Job POST

Creates one or more new jobs.

Request

Method	URL
POST	/api/v1/job/

Parameters

<table>
<tr><th>Type</th><th>Params</th><th>Values</th></tr>
<tr><td>HEAD</td><td>Authorization:</td><td>Basic Username:Password</td></tr>
<tr><td>HEAD</td><td>Content-Type:</td><td>application/json</td></tr>
<tr><td>BODY</td><td colspan="2"><pre>{
    "Jobs": [
        {
            "reference": "ab1234",
            "title": "Senior Kitchenhand",
            "salaryDisplay": true,
            "salaryLowerLimit": 17,
            "salaryUpperLimit": 20,
            "salaryType": "hour",
            "description": "<p>Must have strong experience as kitchenhand Must have strong experience as kitchenhand Must have strong experience as kitchenhand</p>",
            "professionId": 15,
            "locationId": 5379,
            "educationId": 2,
            "skills": [
                5043
            ],
            "languages": [
                1,
                5,
                8
            ],
            "certificates": [
                3145
            ],
            "type": "casual",
            "distance": 50,
            "vehicle": false,
            "applicationUrl": null,
            "applicationEmail": null
        }</pre></td></tr>
</table>

	] }

Response

Status	Response

200	{
	    "success": true,
	    "message": "1 job(s) created.",
	    "details": [
	        {
	            "jobId": 42652,
	            "reference": "ab124",
	            "message": null
	        }
	    ]
	}

200	{
	    "success": false,
	    "message": "0 job(s) created.",
	    "details": [
	        {
	            "jobId": 0,
	            "reference": null,
	            "message": "Invalid job reference or already in use."
	        }
	    ]
	}

Job PUT

Updates one or more jobs

Request

Method	URL
PUT	/api/v1/job/

Parameters

Type	Params	Values
HEAD	Authorization:	Basic Username:Password
HEAD	Content-Type:	application/json

BODY	{
	    "Jobs": [
	        {
	            "jobId": 41658,
	            "title": "Senior Kitchenhand",
	            "description": "<p>Must have strong experience as kitchenhand Must have strong experience as kitchenhand Must have strong experience as kitchenhand</p>"
	        }
	    ]
	}	

Examples

Method	URL	Result
POST	/api/v1/job/	Update one or more jobs

Response

Status	Response

200	{
	    "success": true,
	    "message": "1 job(s) updated.",
	    "details": [
	        {
	            "jobId": 42658,

	"reference": "ab1242",
	            "success": true,
	            "message": "Job updated."
	        }
	    ]
	}

200	{
	    "success": false,
	    "message": "0 job(s) updated.",
	    "details": [ ]
	}

Job DELETE

Closes a specific job

Request

Method	URL
`DELETE`	`/api/v1/job/:id`

Parameters

Type	Params	Values
`HEAD`	`Authorization:`	`Basic Username:Password`
`HEAD`	`Content-Type:`	`application/json`
`URL`	`:id`	`JobId`

Examples

Method	URL	Result
`DELETE`	`/api/v1/job/123`	`Close job 123`

Response

Status	Response

`200`	{
	    "success": true,
	    "message": "Jobs found.",
	    "Jobs": [
	        {
	            "businessId": 21224,
	            "jobId": 40619,
	            "jobTitle": "Front End Web Developer",
	            "createdAt": "2014-07-15T23:36:55.66",
	            "closedAt": null,
	            "salaryDisplay": false,
	            "salaryLowerLimit": 60000,
	            "salaryUpperLimit": 80000,
	            "salaryType": "annum",
	            "description": "<p>We are seeking the services of a top notch front end web developer.</p>

```
<p> You will be working on a high profile website, cutting up
designs from the UI Designer and ensuring that they work across
multiple devices. You must have a lot of experience in the
following</p>
<ul>
<li>CSS</li>
<li>HTML</li>
<li>Javascript</li>
<li>jQuery</li>
<li>Bootstrap</li>
</ul>
<p>If you are interesting in a great company culture in a very
agile environment, please apply</p>",
            "professionId": 161,
            "locationId": 494,
            "educationId": 1,
            "skills": [
                725,
                899,
                2651,
                2732,
                4263,
                8833
            ],
            "languages": [
                1
            ],
            "certificates": [ ],
            "jobType": "permanent",
            "distance": 50,
            "vehicle": false,
            "activationDays": null
        }
    ]
}
```

200

```
{
    "success": false,
    "message": "No jobs found.",
    "Jobs": [ ]
}
```

Fetch GET

Return a list of required data for the given authenticated account

Request

Method	URL
GET	/api/v1/fetch/:object

Parameters

Type	Params	Values
HEAD	Authorization:	Basic Username:Password
HEAD	Content-Type:	application/json
URL	:object	Name of Object see examples

Examples

Method	URL	Result
GET	/api/v1/fetch/profession	Return professions
GET	/api/v1/fetch/category	Return categories
GET	/api/v1/fetch/location	Return locations
GET	/api/v1/fetch/education	Return education levels
GET	/api/v1/fetch/language	Return languages
GET	/api/v1/fetch/certificate	Return certificates
GET	/api/v1/fetch/certificate/39	Return certificates for professionId 39
GET	/api/v1/fetch/skill	Return skills
GET	/api/v1/fetch/skill/40	Return skills for professionId 40

Response

/api/v1/fetch/profession

<table>
<tr><th>Status</th><th>Response</th></tr>
<tr><td>200</td><td>

```
[
    {
        "professionId": 1,
        "name": "Other",
        "categoryId": 1
    },
    {
        "professionId": 2,
        "name": "Accommodation and Hospitality Manager",
        "categoryId": 1
    },
    {
        "professionId": 3,
        "name": "Accounts Officers/Clerks",
    }
]
```

</td></tr>
</table>

NOTE: This is an abbreviated example for profession only. The actual list contains more data. All fetch methods other than skills return the same structure with the relevant objectId name.

NOTE: Skills & Certificates returns all or for the specified profession.

Response

`/api/v1/fetch/skill`

Status	Response

200	[
	{
	"skillId": 7437,
	"name": "Animal Behaviour Correction"
	},
	{
	"skillId": 6825,
	"name": "Animal Care"
	},
	{
	"skillId": 9273,
	"name": "Animal Diets"
	},
	{
	"skillId": 6679,
	"name": "Animal Husbandry"
	},
	{
	"skillId": 4997,
	"name": "Animal Surgery Assistance"
	},
	{
	"skillId": 3362,
	"name": "Aquascape Maintenance"
	}
	]

NOTE: This is an abbreviated example only.

Glossary

Status Codes

All status codes are standard HTTP status codes. The below ones are used in this API.

`2XX` - Success of some kind
`4XX` - Error occurred in client’s part
`5XX` - Error occurred in server’s part

Status Code	Description
200	OK
201	Created
400	Bad request
401	Unauthorized
405	Method not supported
500	Internal Server Error
501	Not Implemented
503	Service Unavailable