UNIWeb API

Overview

The purpose of the API is to integrate UNIWeb with other systems within your organization. The Authenticated API provides secure read/write access to information stored by UNIWeb, and it provides a mechanism to reduce the need to duplicate data.

The UNIWeb API provides:

- An interface that allows you to control who has access to your institution's data through our API.
- A means by which to securely read and update your institution's information.
- Rich data in simple, straightforward JSON for maximum readability and reusability.
- The choice to pre-filter the requested data, to obtain just the subset of information in which you are interested.

The UNIWeb API uses Internet Engineering Task Force (IETF) open authentication standard OAuth 2.0, for authentication and authorization using the Resource Owner Password Credentials Grant protocol. In this protocol, only users with granted permission can access the API.

The following four steps are required to access a resource through API:

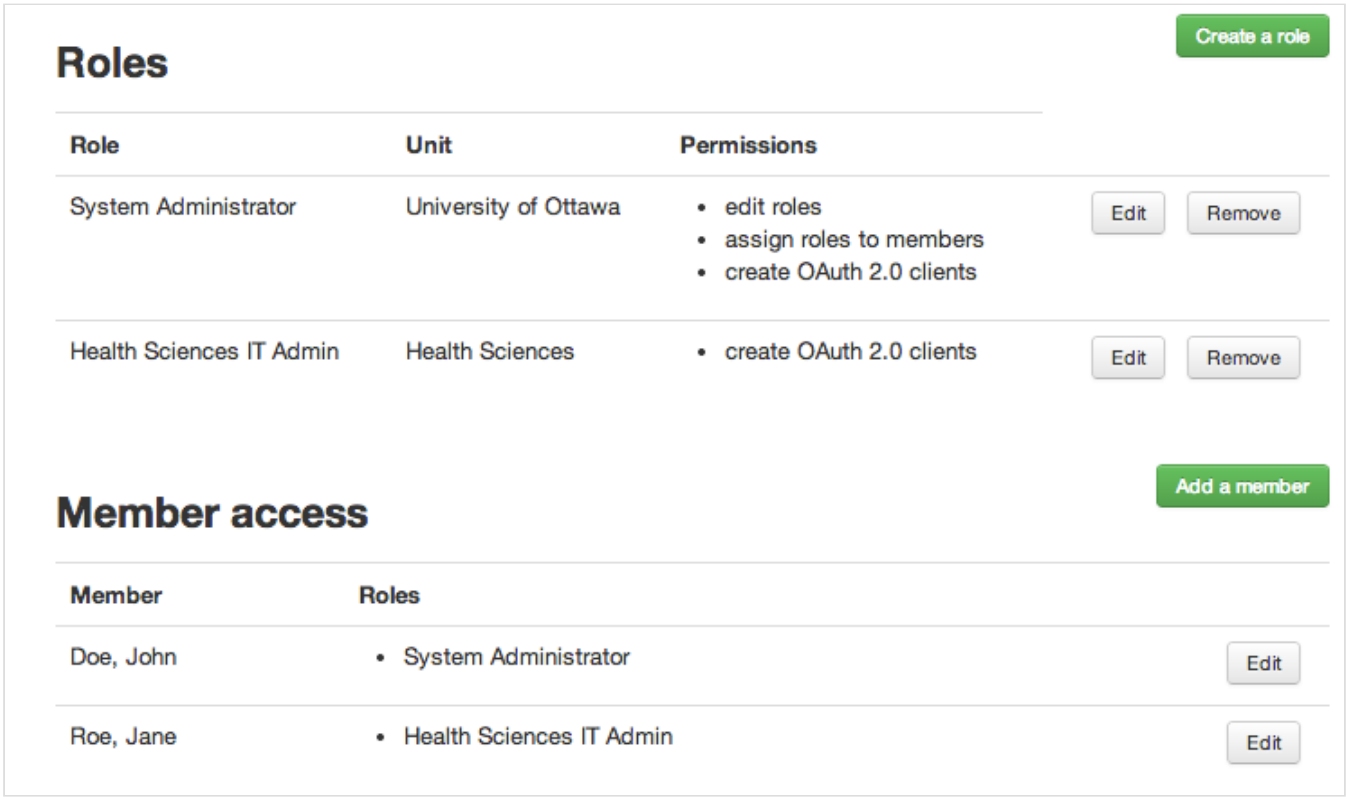
1. Get permission to create OAuth 2.0 Clients
2. Create a OAuth 2.0 client and obtain client credentials
3. Use your Oauth 2.0 client credentials to retrieve an access token.
4. Use your access token to interact with the API. Your token is valid for an hour from the time it is issued.

These steps are explained in more details below.

Setting up Authorized Clients

1. Get permission to create OAuth 2.0 Clients

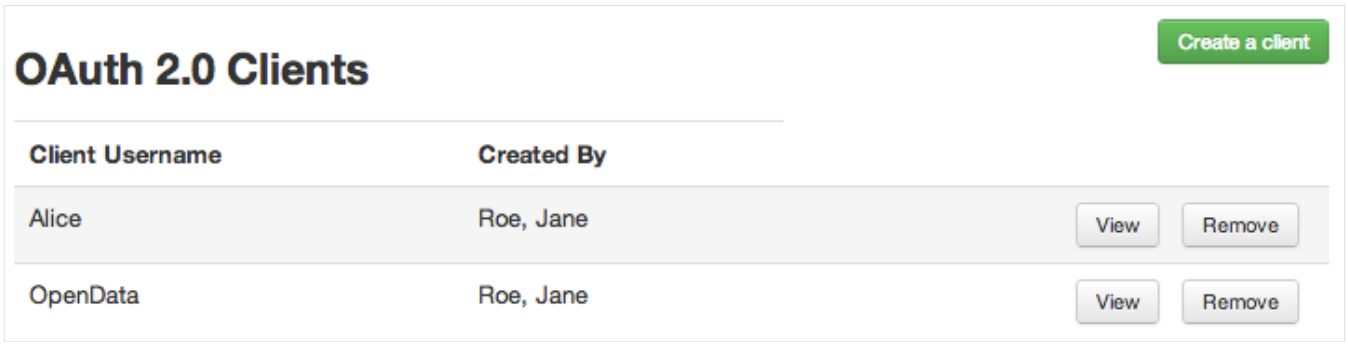
A *System Administrator*, can grant any user permission to *create OAuth 2.0 clients*. If you are not the *System Administrator* yourself, ask the *System Administrator* to give you this permission, as example below shows:



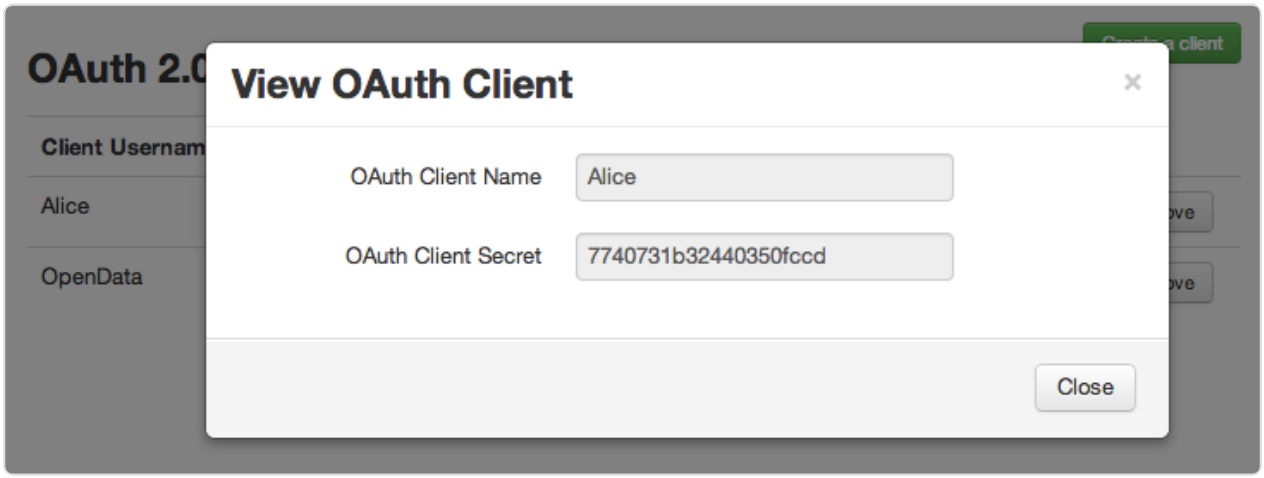
Example above shows *Access Control* page, accessible from *Administration* panel, to *John Doe*, the *System Administrator*. In this example, role *Health Science IT Administrator* has the permission to *create OAuth 2.0 clients* for *Health Sciences* department. *John Doe* assigns this role to *Jane Roe*. *Jane Roe* can now *create OAuth 2.0 clients*.

2. Create a OAuth 2.0 client and obtain client credentials

Using the UNIWeb Interface, you can create, edit, view and remove OAuth 2.0 clients. Each client has a unique username referred to as *Client ID*, and a system generated random password, referred to as *Client Secret*. Example below shows *Jane Roes*'s *OAuth 2.0 Administration* page.



In this example, *Jane Roe* has created two OAuth 2.0 clients. Clicking on the *view* button for *Alice* reveals her*Client Secret* as shown below:



In this hypothetical case, *Alice*'s *Client ID* is *Alice* and her *Client Secret* is *7740731b32440350fccd*. These credentials are used in the next step to authenticate *Alice*.

3. Authenticate and get an Access Token

With the client credentials obtained in step 3, you can authenticate to the UNIWeb *Token Endpoint*, and get an*Access Token*. An *Access Token* is valid for one hour, and it will be used in the next step to retrieve resources from UNIWeb *Resource Endpoint*.

With these pieces of information you will be allowed to make API requests. To do so, you can use one of our pre-built client libraries

- PHP client lib
- Java client lib
- cURL client lib

or you can write your own by following the information in the API In Depth page.

4. Access information through structured requests

API requests are made by submitting JSON objects to the server. They tell the server which action, resources, sections and fields are desired and what filters to apply. In particular, the request objects can have the following properties: **action**, **content**, **id**, **filter**, and **resource**.

Example request object:

```
{
    "action": "read",
    "content": "members",
    "id": "bob@mail.ca",
    "resource": [
        "profile/biography",
        "profile/selected_degrees"
    ]
}
```

Example response for the above request:

```
{
    "profile/biography": "Bob always knew he would be a great scientist",
```

```
    "profile/selected_degrees": [
        {
            "degree_name": "PhD",
            "organization": "McGill University",
            "specialty": "Materials Engineering"
        },
        {
            "degree_name": "Engineering",
            "organization": "University of Ottawa"
        }
    ]
}
```

API Requests

Before requesting information from UNIWeb, it is necessary to understand the terminology used to identify pieces of data stored in the system. The information within a UNIWeb page is usually divided into sections, sub-sections, sub-subsections and so on. A *section* contains a list of *items*. An item within a section is made out *fields*. An API*request* is the mechanism for obtaining the *field values* of all items within a section.

Resource Paths

In UNIWeb, a resource is always associated to a type of *content*. Current content types are: 'members', 'units' and 'groups'.

To request a resource, it is necessary to provide a path to it within UNIWeb. A *request path* can be specified as a string by separating each element in the path with '/'. The path must have the following form:

`page/section/section/section/...`

`page`	The 'page' where the information is displayed within UNIWeb. For example, 'profile', 'cv' or 'graph'.
`section/...`	Sequence of section, subsection, sub-subsection,... that contain the target set of items to retrieve.

For example, the string `cv/education/degrees` refers to all the items within the section *Degrees*, which is a subsection of the *Education* section in the CV page of UNIWeb *members*.
Optionally, a request path can be specified as a JSON object. In particular, this is needed if one desires to request only a subset of the field values of an item. In this case, the *resource path* can be given as

```
{"page/section/section/section/...": ["field name A", "field name B", ...]}
```

It is also possible to encode the entire path as a JSON object. This is useful when requesting multiple sections under a common parent section or page:

```
"page": {
```

```
    "parent_section":[
      "child_section A",
      "child_section B"
    ]
  }
```

The *resource path* above is equivalent to specifying two separate resource paths as strings:

```
  [
    "page/parent_section/child_section A",
    "page/parent_section/child_section B"
  ]
```

Naming Conventions

The names of sections and fields used by the API are derived from the **English titles** of their respective sections and fields shown in the UNIWeb UI. Spaces, slashes and question marks are not allowed in resource names. In addition, resource names are always lowercased. To "normalize" a string to meet API rules, do the following:

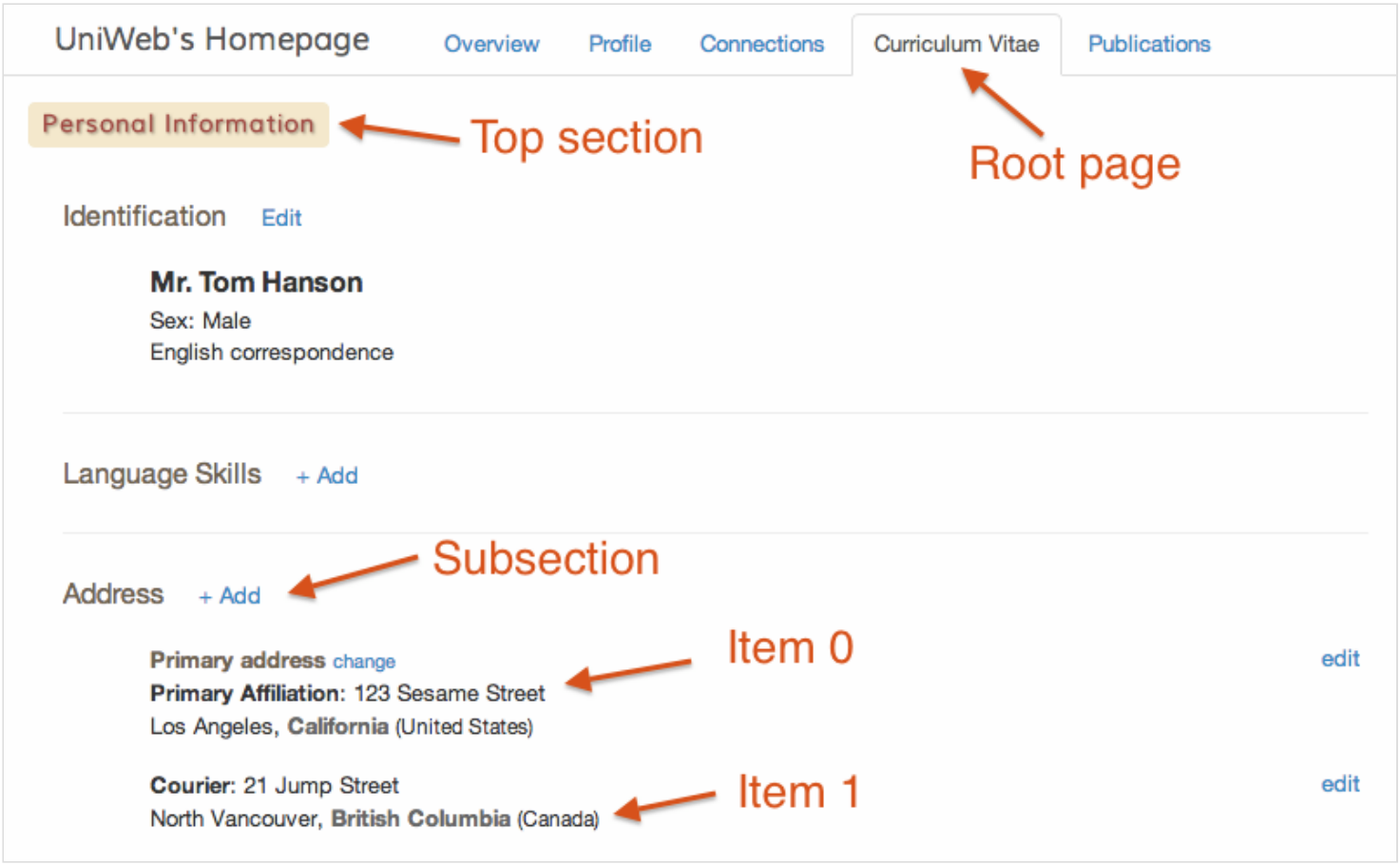
- Lowercase the given string
- Replace the substrings " / ", "/", and " " with "_"
- Replace the substrings "?" with the empty string ""

For example, the string "Postal / Zip Code" is normalized to "postal_zip_code".

Section Names

The names of sections in resource paths must: (1) correspond to the sections names shown in the UNIWeb UI, and (2) be normalized according to the API naming rules described above. For example, the path to the Address resource in a CV is written as

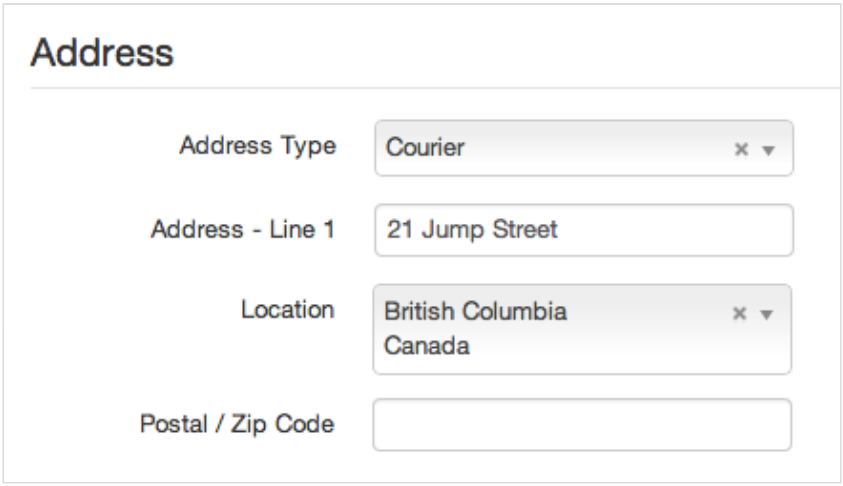
`cv/personal_information/address`



Fields Names

The names of fields in resource paths must: (1) correspond to the field labels in the UNIWeb UI, and (2) be normalized according to the API naming rules described above. For example, the fields shown below in the Address section can be requested as

```
["address_type", "address_-_line_1", "location", "postal_zip_code"]
```



Tip

In the response to an API request, a section item is encoded as a map of field names to field values.

Stucturing Requests

API requests are given as JSON objects with one or more of the following properties.

`token`	required The hashed value returned by the authorization server.
`action`	required A string value specifying the desired action to take. Show options
`content`	required Selects the type of content to retrieve. Show options
`id`	optional The ID of one particular content to request. Show example
`resource`	required One or more paths to the requested resources. Show options
`filter`	optional An object value with filering settings. Show options
`index_by`	optional Selects how the response indexed the resources in the answer. Show options
`language`	optional Responses use the default institution's language unless specified otherwise. Show options

Tip

When requesting items of a section, always expect to receive an array of items even if the target section cannot have multiple items. For example, the `Identification` section in a CV can only have a single item given that it lacks a +Add link (i.e., it has an Edit link instead). Regardless, the response to the resource `"cv/personal_information/identification"` will be an array of items of length one.

Example Requests

Simple Single Resource Read Request

The request that follows would return the public profile information of all people in the Department of Civil Engineering as JSON.

```
{
    "token": "access token",
    "action": "read",
    "content": "members",
    "filter": {
        "unit": "Civil Engineering"
    },
    "resource": "profile"
}
```

Requesting to Read Multiple Resources in a Single Request

The request that follows would return two resources belonging to the user with login name `john@smith.ca`, which include:

- the publicly available research interest tags found on his Profile
- the Degree Name, Specialization, and Thesis Title fields from his CV found under Education > Degrees

```
{
    "token": "access token",
    "action": "read",
    "content": "members",
    "language": "fr",
    "filter": {
        "unit": "McGill",
        "title": "Professor",
        "login": "john@smith.ca"
    },
    "resource": [
        "profile/research_interests",
        {
            "cv/education/degrees": [
                "degree_name",
                "specialization",
                "thesis_title"
            ]
        }
    ]
}
```

Error Messages

Errors will give information about what went wrong with a corresponding request. They will be of the following form:

```
{
    "error": {
        "message": "Error validating access token.",
        "type": "OAuthException",
        "code": 98,
        "error_subcode": 223
    }
}
```

Tip

If you make a request that would include in the response a resource that you do not have access to, the whole request will fail. For example, if you request the CV's of several researchers but some of them belong to a unit to which you do not have permissions to view, the request will fail.