**Members Service Api Documenation (Members area)**

**Descritipon :**

Handles the most basic member actions such as retrieving profileids etc

**Help URL :**

[http://173.160.122.195/Anewluv.Web.MembersService/MembersService.svc/Rest/help](http://localhost/Shell.MVC2.Web.MembersService/MembersService.svc/Rest/help)

**AuthRequired: Username password and Api Key**

\*\* request a correct API key from the service admins \*\*

A user authorization header and API key are needed in the header

Example of the authorization header for a user with **username** : case **password** : drive333

1.)Combine the username and password with a “:” separator - **> case:drive333**

2.) Url encode the combined string as base 64 : Y2FzZTpkcml2ZTMzMw==

i.e url for testing encoding of passwords

<http://www.motobit.com/util/base64-decoder-encoder.asp>

3.) Add the encoded URL to the header

Two headers are required one named “apikey” and one named “Authorization”

a)Sample of api key and authentication header using fiddler :

User-Agent: Fiddler

Content-Type: text/json

Host: 173.160.122.195

Authorization: Y2FzZTpkcml2ZTMzMw==

apikey: 460ad6f3-8216-469f-9b1c-52cffa5d812c

b) Sample of Ajax calls with header included:

<http://wilbloodworth.com/add-request-headers-in-ajax-http-post-using-jquery/>

**Service Call Details and Documenation**

Most of the models that pass values use this viewmodel to hold the properties :

{

"activationcode":"String content",

"email":"String content",

"openididentifier":"String content",

"openidprovider":"String content",

"password":"String content",

"photoid":"1627aea5-8e0a-4371-9022-9b504344e724",

"profileid":2147483647,

"screenname":"String content",

"securityanswer":"String content",

"securityquestion":"String content",

"sessionid":"String content",

"username":"String content"

}

The following details of each call shows the properties required for each call on a call by call basis.

1)**getprofileidbyusername** – gets the profileID by username

**Help url**

[**http://173.160.122.195/Anewluv.Web.MembersService/MembersService.svc/Rest/help/operations/getprofileidbyusername**](http://localhost/Shell.MVC2.Web.MembersService/MembersService.svc/Rest/help/operations/getprofileidbyusername)

**url**

[http://173.160.122.195/Anewluv.Web.MembersService/MembersService.svc/Rest/getprofileidbyusername](http://localhost/Shell.MVC2.Web.MembersService/MembersService.svc/Rest/getprofileidbyusername)

Json Body

{

"screenname":"String content"

}

Screen name is required

Result

The profileID is returned as an ID i.e 45 etc.

2)**getprofileidbyscreenanme** – gets the profileID by screenname

**Help url**

[**http://173.160.122.195/Anewluv.Web.MembersService/MembersService.svc/Rest/help/operations/getprofileidbyscreenname**](http://localhost/Shell.MVC2.Web.MembersService/MembersService.svc/Rest/help/operations/getprofileidbyscreenname)

**url**

[http://173.160.122.195/Shell.MVC2.Web.MembersService/MembersService.svc/Rest/getprofileidbyscreenname](http://localhost/Shell.MVC2.Web.MembersService/MembersService.svc/Rest/getprofileidbyscreenname)

Json Body

{

**"username":"myusername"**

}

Username is required

Result

The profileID is returned as an ID i.e 45 etc.

2)**getprofileidbyusername** – gets the profileID email and profile id

**Help Url**

[http://173.160.122.195/Shell.MVC2.Web.MembersService/MembersService.svc/Rest/help/operations/getprofileidbyopenid](http://localhost/Shell.MVC2.Web.MembersService/MembersService.svc/Rest/help/operations/getprofileidbyopenid)

**url**

[http://173.160.122.195/Anewluv.Web.MembersService/MembersService.svc/Rest/getprofileidbyopenid](http://localhost/Shell.MVC2.Web.MembersService/MembersService.svc/Rest/getprofileidbyopenid)

Json Body

{

**"email":"String content",**

"openididentifier":"String content",

**"openidprovider":"String content",**

}

\*\* email and openidprovider are required \*\*

The email address should be retrived from the openID provider and the open id proivider can be one of the follow values : facebook, yahoo or google. The open id identifier is the key provided by the open id provider that maps to that user;s profile on the open id provider. For now

Result:

The profileID is returned as an ID i.e 45 etc.

5)**emailaddress validation**

- [http://173.160.122.195/MembersService/MembersService.svc/Rest/checkifemailalreadyexists](http://localhost/Shell.MVC2.Web.MembersService/MembersService.svc/Rest/checkifemailalreadyexists/%7bEMAILADDRESS%7d)

sample JSON body

{

"email":"ola\_lawal@yahoo.com"

}

-returns a **true** if it exists i,e email address is invalid

5)**screennamevalidation**

[http://173.160.122.195/MembersService/MembersService.svc/Rest/checkifscreennamealreadyexists](http://localhost/Shell.MVC2.Web.MembersService/MembersService.svc/Rest/checkifscreennamealreadyexists/%7bSCREENAME%7d)

{

"screenname":"myname",

}

-reutrns **true** if screen name already exists -user must pick a new one

5)**username**

must be at least six chars , can contain special chars within reason

[http://173.160.122.195/MembersService/MembersService.svc/Rest/checkifusernamealreadyexists](http://localhost/Shell.MVC2.Web.MembersService/MembersService.svc/Rest/checkifusernamealreadyexists/%7bUSERNAME%7d)

{

"username":"firstuser"

}

- returnes **true** if usenrame already exists- user must pick a new one