**Authentication and MembershipService Api Documenation**

**Descritipon :**

Handles authentication of users into the Anewluv application

**Help URL :**

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

**AuthRequired: Api Key**

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

**AuthRequired: API key only**

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

API key are needed in the header

Sample Fiddler header with API key included

User-Agent: Fiddler

Content-Type: text/json

Host: 173.160.122.195

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

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)**validateuserbyusernamepassword** – validates the user using basic unecrpted user name and password information.

**Help url** :

[http://173.160.122.195/Shell.MVC2.Web.AuthenticationService/MembershipService.svc/Rest/help/operations/validateuserbyusernamepassword](http://localhost/Shell.MVC2.Web.AuthenticationService/MembershipService.svc/Rest/help/operations/validateuserbyusernamepassword)

**Url** :

[http://173.160.122.195/Shell.MVC2.Web.AuthenticationService/MembershipService.svc/Rest/validateuserbyusernamepassword](http://localhost/Shell.MVC2.Web.AuthenticationService/MembershipService.svc/Rest/validateuserbyusernamepassword)

Json Body

{

**"username":"myusername",**

**"password":"String content"**

}

Username and password is required

Result:

Returns **True** if user is validated or **False** if not

2)**validateuserbyopenid** – validates the user using open id info

**Help url** :

http://173.160.122.195/Shell.MVC2.Web.AuthenticationService/MembershipService.svc/Rest/help/operations/validateuserbyopenid

**Url** :

[http://173.160.122.195/Shell.MVC2.Web.AuthenticationService/MembershipService.svc/Rest/validateuserbyopenid](http://localhost/Shell.MVC2.Web.AuthenticationService/MembershipService.svc/Rest/validateuserbyopenid)

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:

Returns **True** if user is validated or **False** if not

2)**create user** –

<http://173.160.122.195/Anewluv.Web.AuthenticationService/MembershipService.svc/Rest/createuser>

here is a sample of the POST JSON data needed for the createprofile

{

"birthdate":"\/Date(928167600000-0500)\/",

"city":"Minneapolis",

"country":"United States",

"email":"testdsdsdasda@yahoo.com",

"gender":"Male",

"isApproved":true,

"lattitude":1.26743233E+15,

"longitude":1.26743233E+15,

"openidIdentifer":"",

"openidProvidername":"",

"password":"kayode02",

"providerUserKey":{},

"screenname":"testofusername23",

"securityAnswer":"",

"securityQuestion":"",

"stateprovince":"Minnesota",

"status":0,

"username":"examplecreate",

"zippostalcode":"55411"

}

For now password is sent in plain text but will be over SSL later

\*\*\*NEW \*\*\*\*

I added a new response object that returns JSON data in the following format :

{

"Documents":[{

"document":[81,

109,

70,

122,

90,

83,

65,

50,

78,

67,

66,

84,

100,

72,

74,

108,

89,

87,

48,

61],

"documenttype":0,

"filename":"String content",

"mimetype":"String content"

}],

"ResponseMessages":[{

"dataelement":"String content",

"errormessage":"String content",

"message":"String content"

}],

"currentstatuscode":0,

"currentstatusdate":"\/Date(928167600000-0500)\/",

"email":"String content",

"profileid1":"String content",

"profileid2":"String content",

"profilestatus":0,

"profilestatusdate":"\/Date(928167600000-0500)\/",

"requestreturnflag":true,

"responsecreatedate":"\/Date(928167600000-0500)\/"

}

ResponseMessages will contain any error or other messages regarding if the profile was created or activated or any other type of action. IF there were any problems the errormessage field will be populated as well.

Also

The createuser service call validates unequie email address and username only, in the future we will validate long and lat to make sure they match city and country :

For eample a duplicate email will return the following JSON

{

"Documents":[],

"ResponseMessages":[{"dataelement":"","message":"","errormessage":"Duplicate email : the email :testdsdsdasda@yahoo.comalready exists"}],

"currentstatuscode":0,"currentstatusdate":"\/Date(-62135575200000-0600)\/","email":"","profileid1":"","profileid2":"","profilestatus":0,"profilestatusdate":"\/Date(-62135575200000-0600)\/","requestreturnflag":false,"responsecreatedate":"\/Date(-62135575200000-0600)\/"}

And duplicate username:

{"Documents":[],"ResponseMessages":[{"dataelement":""," errormessage":"Duplicate username : the username :examplecreatealready exists","message":"Unable to create profile"}],"currentstatuscode":0,"currentstatusdate":"\/Date(-62135575200000-0600)\/","email":"","profileid1":"","profileid2":"","profilestatus":0,"profilestatusdate":"\/Date(-62135575200000-0600)\/","requestreturnflag":false,"responsecreatedate":"\/Date(-62135575200000-0600)\/"}

On success ful created profile :

{"Documents":[],"ResponseMessages":[{"dataelement":""," errormessage":"” message":"profile creatred succesfully"}],"currentstatuscode":0,"currentstatusdate":"\/Date(-62135575200000-0600)\/","email":"","profileid1":"","profileid2":"","profilestatus":0,"profilestatusdate":"\/Date(-62135575200000-0600)\/","requestreturnflag":false,"responsecreatedate":"\/Date(-62135575200000-0600)\/"}

Option B. Facebook or openID user

-pretty much same as above , difference is you use the following JSON payload instead

:

{

"birthdate":"\/Date(928167600000-0500)\/",

"city":"Minneapolis",

"country":"United States",

"email":"testdsdsdasda@yahoo.com",

"gender":"Male",

"isApproved":true,

"lattitude":1.26743233E+15,

"longitude":1.26743233E+15,

"openidIdentifer":"1",

"openidProvidername":"Facebook",

"password":"kayode02",

"providerUserKey":{},

"screenname":"testofusername23",

"securityAnswer":"",

"securityQuestion":"",

"stateprovince":"Minnesota",

"status":0,

"username":"examplecreate",

"zippostalcode":"55411"

}

As you can see the openIDprovider and Identifier are specified in the JSOn payload.

**11)Activiate profile**

**Help URL**

<http://173.160.122.195/Anewluv.Web.AuthenticationService/MembershipService.svc/Rest/help/operations/activateprofile>

**URL**

<http://173.160.122.195/Anewluv.Web.AuthenticationService/MembershipService.svc/Rest/activateprofile>

Request JsonBody

{

"Country":"String content",

"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"

}

**Notable items**

Activationcode – required

Profileid -required

**Sample call**

**URL**

<http://173.160.122.195/Anewluv.Web.AuthenticationService/MembershipService.svc/Rest/activateprofile>

**JsonBody**

{

"activationcode":"4359459435043543543",

"profileid":1,

}

**Result**

True if profile successfully activated

**11)Recover Activaition code – retrives an activation code**

**Help Url**

<http://173.160.122.195/Anewluv.Web.AuthenticationService/MembershipService.svc/Rest/recoveractivationcode>

**URL**

<http://173.160.122.195/Anewluv.Web.AuthenticationService/MembershipService.svc/Rest/recoveractivationcode>

**JsonBody**

Request JsonBody

{

"Country":"String content",

"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"

}

**Notable items**

**emailaddress -required**

profileid –optional

**Sample call**

<http://173.160.122.195/Anewluv.Web.AuthenticationService/MembershipService.svc/Rest/activateprofile>

**JsonBody**

{

"emailaddress":"**ola\_323@yahoo.com**",

"photostatus":true,

"photouploadviewmodel":null,

}

**only the email address is required for this call.**

**Result**

String activation code is result