API AUTHENTICATION

MEETLINKSHARE

Version 1.0

Confidentiality and Non-discloser Notice

Document Information and Approvals

Version History

Version #	Date	Revised By	Reason for change
1.0	22-07-2011	Priya/Senthil	*Draft version*

Document Approvals

Approver Name	Project Role	Signature/Electronic Approval	Date

TABLE OF CONTENTS

General Information

Purpose

This document is written for application developers using the MLS Data APIs to access a user's data. This enables third-party developers to build applications that can fetch data on behalf of MLS users without the need to first obtain their usernames and passwords. MLS uses the Oauth for the API authentication.

Scope

This document explains the procedure involved in authenticating the API calls using OAuth.

Benefits

OAuth is open protocol to allow secure API authorization in a simple and standard method from desktop and web applications. In short, it means that a user of your service can provide you limited access to a third party account of theirs. OAuth is often described as a valet key that your users can give you to access their accounts on other services. OAuth to provide limited access to account permissions or information to third-party tools (web or client).OAuth provides a standard protocol for negotiating secure access tokens, which don't reveal usernames or passwords to the third-party tool

Authentication

This Authentication protocol is an extension of the OAuth 1.0 protocol which enables websites or applications (Consumers) to access Protected Resources from a web service (Service Provider) via an API, without requiring Users to disclose their Service Provider credentials to the Consumers. More generally, OAuth creates a freely-implementable and generic methodology for API authentication.

This extension's intent is to provide a pattern that will support the usage concepts of "Trusted" consumer applications, Multi-Tenant / multi-Users with multi-personas. As well as provide a token based trust for applications without web based end points.

OAuth Core 1.0 references and definitions can be viewed : *http://oauth.net/core/1.0/*

TERMINOLOGY

Service Provider

A web application that allows access via OAuth.

User

An individual who has an account with the Service Provider.

Consumer

A website or application that uses OAuth to access the Service Provider on behalf of the User.

Protected Resource

An access-restricted resource that can be obtained from the server using an OAuth-authenticated request.

Resource owner

An entity capable of accessing and controlling protected resources by using credentials to authenticate with the server.

Credentials

Credentials are a pair of a unique identifier and a matching shared secret. OAuth defines three classes of credentials: client, temporary, and token, used to identify and authenticate the client making the request, the authorization request, and the access grant, respectively.

Token

A unique identifier issued by the server and used by the client to associate authenticated requests with the resource owner whose authorization is requested or has been obtained by the client. Tokens have a matching shared-secret that is used by the client to establish its ownership of the token, and its authority to represent the resource owner.

TYPES OF CONSUMERS

MLS applications

MLS iPad or web application.

3rd Party Consumer Applications

Defined as applications written and made public for consumption across Tenants.

AUTHENTICATION PROTOCOLS

- **MLS iPad/Web application** ***: Credentials based***

- **3rd Party Consumer Applications : *Oauth***

CREDENTIALS BASED AUTHENTICATION

PROCEDURE FOLLOWED:

- ✓ MLS Application collects the User's credentials directly
- ✓ MLS Application concatenates the user name and password with a space and base64 encodes the credentials
- ✓ MLS Application puts the encoded credentials in the body of the request (no parameter assignment, just put the bytes in the request)
- ✓ MLS Application posts them to the following URI depending on what user type your using, Example:
    - ○ Portal User: [POST] https://api.meetlinkshare.com/v1/authenticate
    - ○ Weblink User: [POST] https://api.meetlinkshare.com/v1/authenticate
- ✓ The Service Provider will hand the MLS Application back an Access Token via:
    - ○ Response body: access_token=c2VjcmV0IGtleSBmb3Igb2F1dGggdHdpdGVy
    - ○ Header:

      access _token= c2VjcmV0IGtleSBmb3Igb2F1dGggdHdpdGVy
- ✓ The MLS Application will access the User's data using the Access Token

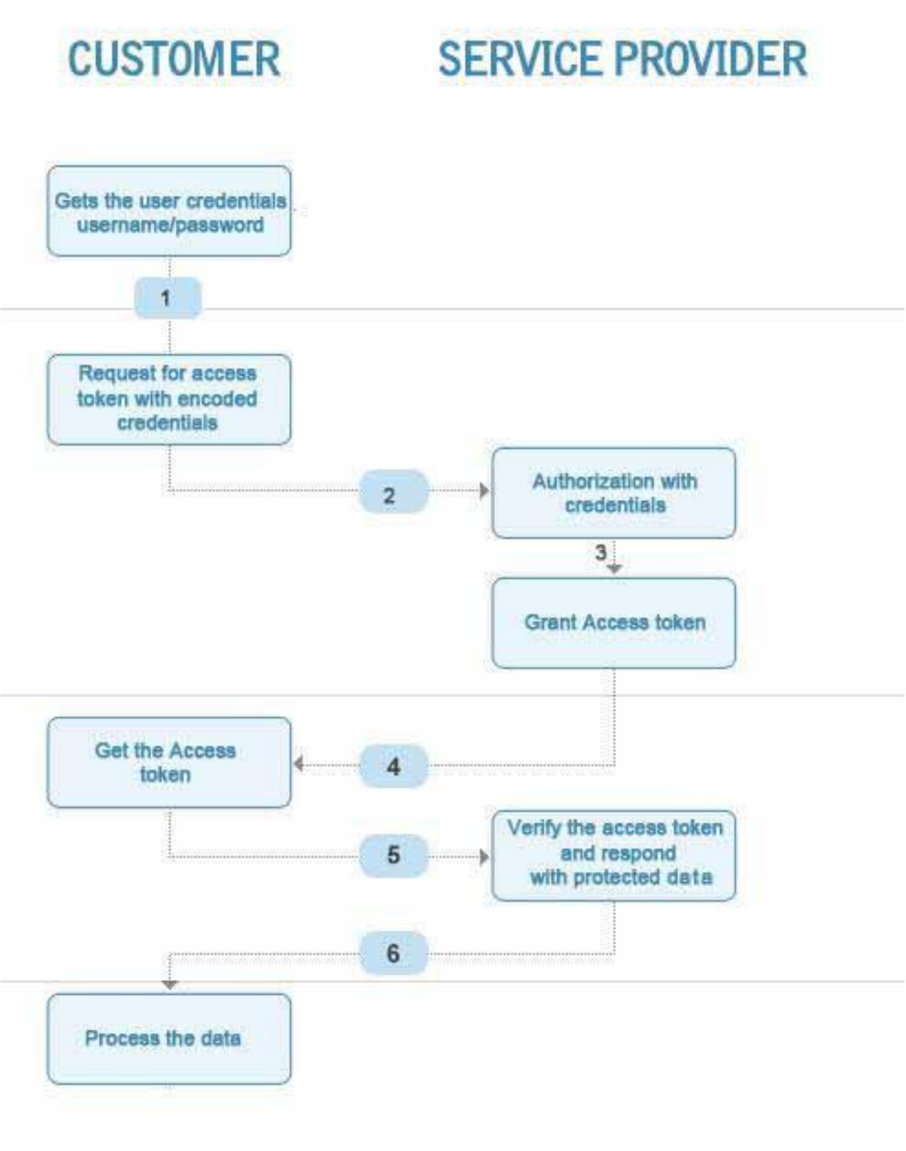
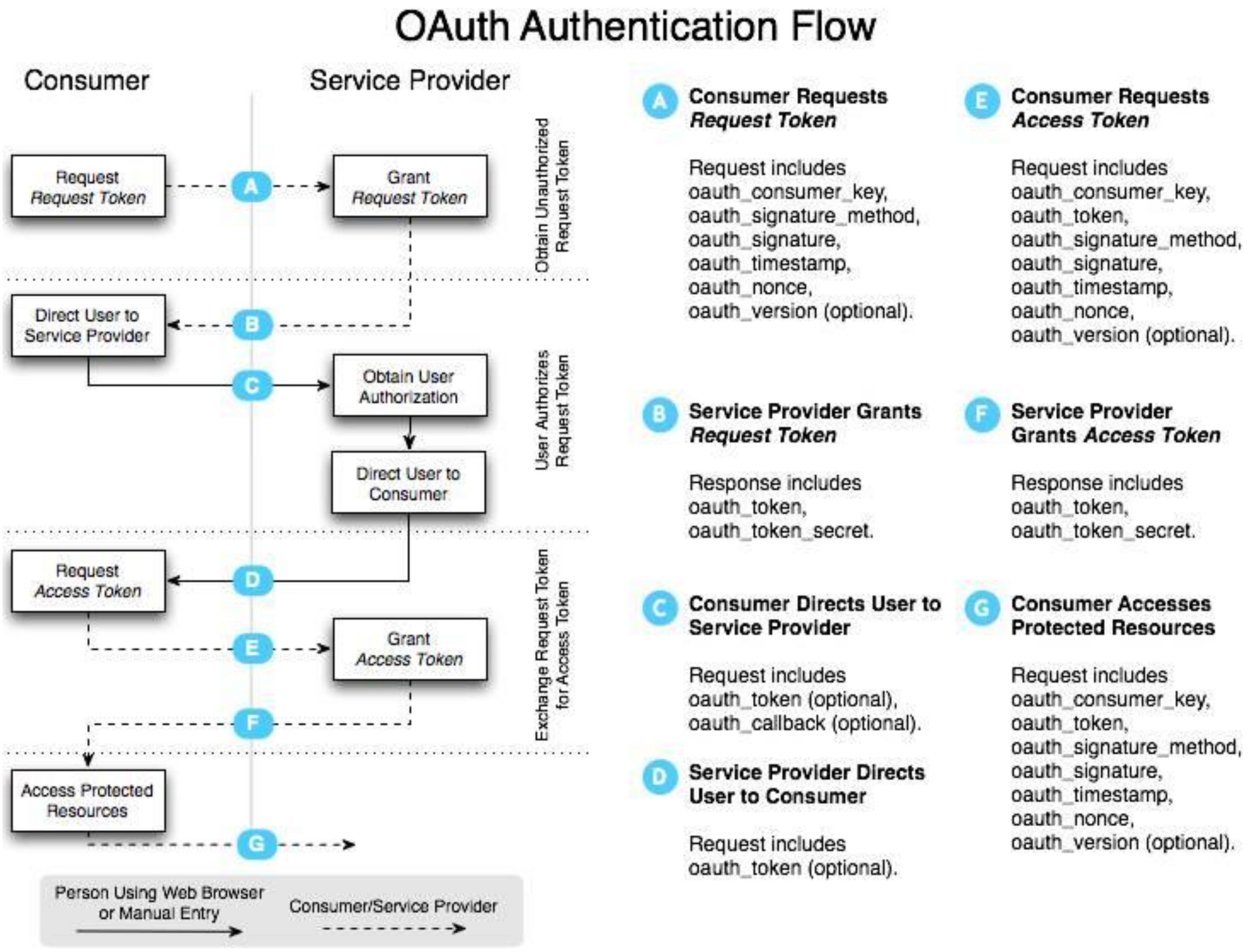

CUSTOMER
SERVICE PROVIDER
Gets the user credentials username/password
1
Request for access token with encoded credentials
2
Authorization with credentials
3
Grant Access token
Get the Access token
4
5
Verify the access token and respond with protected data
6
Process the data

FIG 1.1 Credentials based authentication

OAUTH AUTHENTICATION

PROCEDURE FOLLOWED:

- ✓ Consumer Application registers with the MLS to get the consumer token and consumer secret key
- ✓ Consumer Request the request token with the consumer token and consumer secret key service provider will respond with oauth_key.
- ✓ User is directed to MLS for authorizing with oauth key with the call back url
- ✓ User submits credentials after the authorization the user is redirected to the customer application by passing oauth verifier
- ✓ Consumer application requests for the access token with the request token and oauth verifier
- ✓ Service Provider responds with the access token
- ✓ Consumer access the protected resources with the access token


OAuth Authentication Flow
Consumer
Service Provider
Request *Request Token*
A
Grant *Request Token*
Obtain Unauthorized Request Token
Direct User to Service Provider
B
C
Obtain User Authorization
Direct User to Consumer
User Authorizes Request Token
Request *Access Token*
D
E
Grant *Access Token*
F
Exchange Request Token for Access Token
Access Protected Resources
G
Person Using Web Browser or Manual Entry
Consumer/Service Provider

A Consumer Requests *Request Token*

Request includes oauth_consumer_key, oauth_signature_method, oauth_signature, oauth_timestamp, oauth_nonce, oauth_version (optional).

B Service Provider Grants *Request Token*

Response includes oauth_token, oauth_token_secret.

C Consumer Directs User to Service Provider

Request includes oauth_token (optional), oauth_callback (optional).

D Service Provider Directs User to Consumer

Request includes oauth_token (optional).

E Consumer Requests *Access Token*

Request includes oauth_consumer_key, oauth_token, oauth_signature_method, oauth_signature, oauth_timestamp, oauth_nonce, oauth_version (optional).

F Service Provider Grants *Access Token*

Response includes oauth_token, oauth_token_secret.

G Consumer Accesses Protected Resources

Request includes oauth_consumer_key, oauth_token, oauth_signature_method, oauth_signature, oauth_timestamp, oauth_nonce, oauth_version (optional).

FIG 1.2 OAuth based authentication

OAuth Process flow – sample request and response

ClientApp

consumer/clientapp/register

Register an consumer application	
URI	*consumer/register*
VERB	POST
Request	**Consumer_id:** Login id of the consumer. **client_app_name:** This is a mandatory parameter specifying the client application name. http://api.meetlinkshare.com/v1/consumer/register
Response	**consumer_id**: Consumer's unique id (to be used only for the runtime traffic). **Name**: This the client application name. **Status**: This is the status of the client application. **Customer Key**: This is the key associated with the client application. **Secret key**: This is the secret associated with the key. *Success 2xx with proper response* *<clientapp>* *<consumer>* *<customer_key> unique key with random 32 characters </customer_key>* *<secret> unique key with random 16 characters </secret>* *<loginid>consumer</loginid>* *<clientapplicationname>clientapp_name</clientapplicationname>* *<status>true</status>* *<created_at>2010-02-02 09:18:16 UTC</created_at>* *</consumer>*

	</clientapp> *<clientapp>* *Failure 5xx with proper error message* *<error>* *<message>Couldn't create Client Application. Invalid api group name or loginid</message>* *</error>* *</clientapp>*

consumer/clientapp/delete

Deletes a client application for specified customer	
URI	*/consumer/delete*
VERB	GET
Request	**Consumer_id:** Login id of the consumer (to be used for portal access).This is a mandatory parameter. **Client_app_name:** Specifying the client app name http://api.meetlinkshare.com/v1/consumer/delete?client_app_name=clientapp_name&consumerid=consumer
Response	*Success 2xx with proper response* *Failure 5xx with proper error message* *<consumer><error>* *<message>Consumer or Client Application doesn't exists</message>* *</error>* *</consumer>*

Key

consumer/key/generate

Generate a consumer key and secret for the specified Client	
URI	*/consumer/key/generate*
VERB	GET
Request	**consumerid:** Login id of the consumer **Client_app_name:** The client application name *http://api.meetlinkshare.com/v1/consumer/key/generate?client_app_name=clientapp_name&consumerid=consumer*
Response	**Key:** This is the key generated. **Secret**: This is the secret associated with the key. **application_name** This is the client application name associated with the key. **Status**: This is the status of the key.(true/false) *Success 2xx with proper response* <consumer> <key> unique key with random 32 characters </key> <secret> unique key with random 16 characters </secret> <clientapplicationname>clientapplication_name</clientapplicationname> <status>true</status> <created_at/> </consumer> *Failure 5xx with proper error message* *<consumer>* *<error>* *<message>Key Not approved</message>*

	</error> *</consumer>*

consumer/*key/validate*

Validate the key and return the attributes of the Key	
URI	*/consumer/key/validate*
VERB	GET
Request	**Key:** This is a mandatory parameter. http://api.meetlinkshare.com/v1/consumer/key/validate?key=66181434f9f8dba51d34a8d052f7676b
Response	**Key:** This is the key generated. **Secret**: This is the secret associated with the key. **Client_app_name**:This is the client application name associated with the key. **Status**: This is the status of the key.(true/false) *Success 2xx with proper response* *<consumer>* *<key> unique key with random 32 characters </key>* *<secret> unique key with random 16 characters </secret>* *<clientapplicationname>clientapp_name</clientapplicationname>* *<status>true</status>* *<created_at/>* *</consumer>* *Failure 5xx with proper error message* *<consumer>* *<error>* *<message>Key doesn't Exists</message>* *</error>* *</consumer>*
Failure Scenarios	If key is invalid.

consumer/*key/revoke*

Revokes the Key	
URI	*/consumer/key/revoke*
VERB	GET
Request	**Key:** This is a mandatory parameter. http://api.meetlinkshare.com/v1/consumer/clientapp/key/revoke?key=66181434f9f8dba51d34a8d052f7676b
Response	**Status**: This is the status of the key.(Revoked) *Success 2xx with proper response* *<key>* *<status>Revoked</status>* *</key >* *Failure 5xx with proper error message* *<consumer>* *<error>* *<message>Key doesn't Exists</message>* *</error>* *</consumer>*

consumer/*key/approve*

Approves the Key	
URI	*/consumer/key/approve*
VERB	GET
Request	**Key:** This is a mandatory parameter. http://api.meetlinkshare.com/v1/consumer/clientapp/key/approve?key=66181434f9f8dba51d34a8d052f7676b
Response	**Status**: This is the status of the key.(Approved) *Success 2xx with proper response* *<key>* *<status>Approved</status>* *</key >* *Failure 5xx with proper error message* *<consumer>* *<error>* *<message>Key doesn't Exists</message>* *</error>* *</consumer>*

OAuth

Request_token/generate

Generates the request_token for specified consumer key	
URI	*/request_token/generate*
VERB	GET
Request	**Key:** This is a mandatory parameter. http://api.meetlinkshare.com/v1/requesttoken/generate?key=7b814aef20b0a93f289f926c469ca1c8
Response	*Success 2xx with proper response* *<request_token>* *<token> unique key with random 32 characters </token>* *<secret> unique key with random 16 characters </secret>* *</request_token>* *Failure 5xx with proper error message* *<request_token>* *<error><message>Key is not valid</message>* *</error>* *</request_token>*

Request_token/validate

Validates the request token	
URI	*/request_token/validate*
VERB	GET
Request	**request_token:** This is a mandatory parameter. http://api.meetlinkshare.com/v1/requesttoken/validate?request_token=fd11dc3e39c5b8959cb3e23d7965dee8
Response	*Success 2xx with proper response* *<request_token>* *<token> unique token with random 32 characters </token>*

	<token_secret> unique token_secret with random 16 characters </token_secret> <key> unique key with random 32 characters </key> <clientapp_name>app2</clientapp_name> </request_token> Failure 5xx with proper error message <request_token> <error><message>Request token is not valid </message> </error> </request_token>
Failing Scenarios	If request_token is invalid.

Request_token/revoke

Request token is revoked and status becomes inactive	
URI	/request_token/revoke
VERB	GET
Request	**Request_token:** This is a mandatory parameter. http://api.meetlinkshare.com/v1/requesttoken/revoke?request_token=cf0701c7a47ad33869f1b09ccefbf95f
Response	Success 2xx with proper response n/a Failure 5xx with proper error message <request_token> <error> <message>Request token is not valid</message> </error> </request_token>
Remarks	Update the status of request token as “disabled”.

Request_token/approve

Request token is approved and status becomes active	
URI	*/request_token/approve*
VERB	GET
Request	**Request_token:** This is a mandatory parameter. http://api.meetlinkshare.com/v1/requesttoken/approve?request_token=cf0701c7a47ad33869f1b09ccefbf95f
Response	*Success 2xx with proper response* *n/a* *Failure 5xx with proper error message* *<request_token>* *<error>* *<message>Request token is not valid</message>* *</error>* *</request_token>*
Remarks	*Update the status of request token as "enabled".*

Oauth_verifier/create

creates the verifier	
URI	*/verifier/create*
VERB	GET
Request	**Verifier:** This is a mandatory parameter. **userid:** This is a mandatory parameter. The optional parameters are user_attribute1,user_attribute2 http://api.meetlinkshare.com/v1/verifier/create?verifier=verifier&userid=user_id&attribute1=attr1&user_attribute2=attr2 \
Response	*Success 2xx with proper response* *n/a* *Failure 5xx with proper error message* *<oauth_verifier>* *<error><message>Unable to create Oauth Verifier</message>* *</error>* *</oauth_verifier >*

Oauth_verifier/delete

Purge the verifier	
URI	*/oauth/verifier/delete*
VERB	GET
Request	**Verifier:** This is a mandatory parameter. http://api.meetlinkshare.com/v1/verifier/delete?verifier=verifier
Response	*Success 2xx with proper response* *n/a* *Failure 5xx with proper error message* *<oauth_verifier >* *<error>* *<message>Unable to delete Oauth Verifier</message>* *</error>* *</oauth_verifier >*

Access_token/generate

Generates the accesstoken	
URI	*/accesstoken/generate*
VERB	GET
Request	**Oauth_Verifier:** This is a mandatory parameter. **Request_token:** This is a mandatory parameter. http://api.meetlinkshare.com/v1/accesstoken/generate?request_token=requesttoken&verifier=verifier&key=342b0efe3a491fc575e63d57da97a74b
Response	**Token:** This is the access token generated. **Secret:** This is the secret generated for the particular access token.

	Key: This the key associated with the particular access token. **Status:** This specifies the status of the particular access token(true/false) *Success 2xx with proper response* *<accesstoken>* *<token> unique token with random 32 characters </token>* *<secret> unique secret with random 16 characters </secret>* *<key> unique key with random 32 characters </key>* *</accesstoken>* *Failure 5xx with proper error message* *<accesstoken>* *<error>* *<message>Key is not valid</message>* *</error>* *</accesstoken>*

Access_token/validate

Return the attributes of the access token	
URI	*/access_token/validate*
VERB	GET
Request	**Access_token:** This is a mandatory parameter. **key:** This is an mandatory parameter. http://api.meetlinkshare.com/v1/accesstoken/validate?access_token=cc198acddc70d308bf4900ec0cce0b7c&key=*66181434f9f8dba51d34a8d052f7676b*
Response	**Accesstoken:** This is the access token generated.

	Secret: This is the secret generated for the particular access token. *Success 2xx with proper response* *< accesstoken >* *<token> unique token with random 32 characters </token>* *<token_secret> unique secret with random 16 characters </token_secret>* *<key> unique key with random 32 characters </key>* *<issuedtimestamp>2011-07-25 04:59:53 UTC</issuedtimestamp>* *</accesstoken>* *Failure 5xx with proper error message* *<accesstoken>* *<error>* *<message>Access token is not valid</message>* *</error>* *</accesstoken>*
Failing Scenarios	If access_token is invalid.

Accesstoken /revoke

Access token is revoked and status becomes inactive	
URI	*/accesstoken/revoke*
VERB	GET
Request	**Access_token:** This is a mandatory parameter. http://api.meetlinkshare.com/v1/accesstoken/revoke?access_token=accesstoken
Response	*Success 2xx with proper response* *n/a* *Failure 5xx with proper error message* *<accesstoken>* *<error>* *<message>Access token is not valid</message>* *</error>* *</accesstoken>*
Remarks	*Update the status of access token as "disabled".*

Accesstoken /approve

Access token is approved and status becomes active	
URI	*/accesstoken/approve*
VERB	GET
Request	**Access_token:** This is a mandatory parameter. http://Api.meetlinkshare.com/v1/accesstoken/approve?access_token=accesstoken
Response	*Success 2xx with proper response* *n/a* *Failure 5xx with proper error message* *<accesstoken>* *<error>* *<message>Access token is not valid</message>* *</error>* *</accesstoken>*
Remarks	*Update the status of access token as "enabled".*

Access_token/update

General purpose update method	
URI	*/access_token/update*
VERB	GET
Request	**Access_token:** This is a mandatory parameter. **User_id** : This is a mandatory parameter. The optional parameters are user_attribute1,user_attribute2,user_attribute3,status http://api.meetlinkshare.com/v1/accesstoken/update?access_token=accesstoken &userid=user_id&status=true
Response	*Success 2xx with proper response* *n/a* *Failure 5xx with proper error message* *<accesstoken>* *<error>* *<message>Access token is not valid</message>* *</error>* *</accesstoken>*