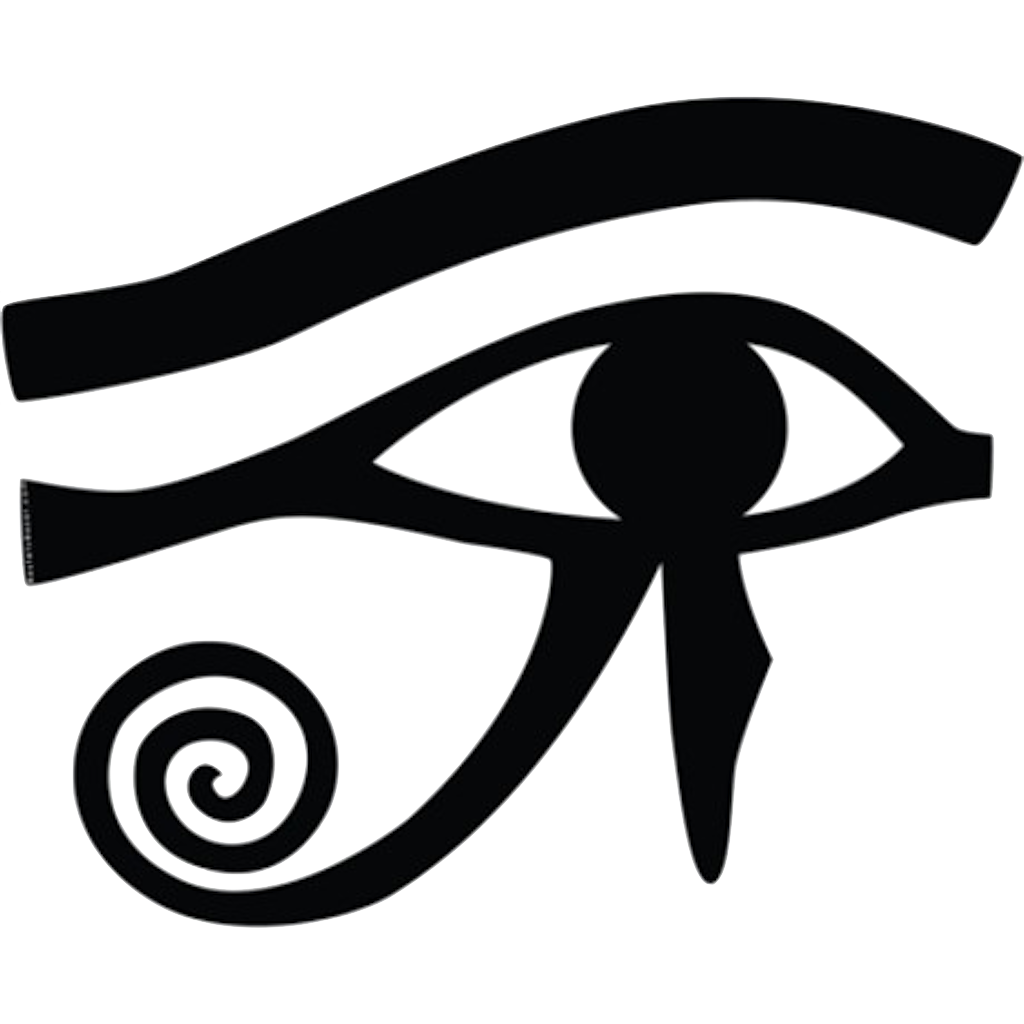
May 29, 2014 – Version 0.1

Open GeoSocial API

User’s Guide



# Table Of Contents

## Introduction

The purpose of this guide is the help data publishers through the step of publishing “Products” that can be used as actionable information by end-users. After following the described steps, “Products” will be discoverable, searchable and shareable through social networks such as Facebook/Twitter.

## Application Registration

The “Application” or publishing service needs an Id and secret that can be used for security and the sharing of data through social networks. We will be using Facebook for this purpose.

As a developer, you will need to sign up for Facebook Account if you do not have one.

Go to [www.facebook.com](https://www.facebook.com/).

If you see the signup form, fill out your name, birthday, gender and email address. If you don't see the form, click **Sign Up**, then fill out the form.

Pick a password.

Click **Sign Up**.

After you complete the signup form, we'll send an email to the address you provided. To complete the signup process, click the confirmation link.

Login and Create a New Application

Go to <https://developers.facebook.com/>

Under Apps menubar tab, select “Create a New App”

Fill out dialog information

Congratulations. You now have a fbAppId and fbAppSecret

Under setting: You also have an Application namespace

Under Open Graph, you can define Object types, Action Types and Stories

Fill this out later….

### Twitter App Registration

<https://apps.twitter.com/>

## Security

### Consumer Application Registration

As a publisher, you may want to force consumer applications to register at your site. This will also allow you to know the owner of the application since he or she will have to login to register the application as well.

Another option is to accept any application that has a valid Facebook Id. If the application does not behave properly, you may choose to ban it from access.

### Security Protocol

[Hawk](https://github.com/hueniverse/hawk) is an HTTP authentication scheme using a message authentication code (MAC) algorithm to provide partial HTTP request cryptographic verification. Consumer applications are required to provide credentials to get access to the products.

Example:

var credentials = {

id: app\_id,

key: app\_secret,

algorithm: 'sha256'

}

Consumer applications will use their Facebook Id and a Facebook access token as a key (rather than provide their secret). An Access Token can be easily obtained by the application from Facebook directly and used for all secured transactions. SHA-256 is the preferred algorithm to use.

The consumer application will also pass the user email as part of an extended field in the authorization header

// Generate Authorization request header

var header = Hawk.client.header('http://example.com:8000/resource/1?b=1&a=2', 'GET', {

credentials: credentials,

ext: ‘pat@cappelaere.com'

});

var options = {

uri: url,

method: 'GET',

headers: {

authorization: header.field

}

};

Request(options, function(err, response, body){

if( !err ) {

var isValid = Hawk.client.authenticate(

response, credentials, header.artifacts, { payload: body });

if( isValid ) {

try{

var json = JSON.parse(body)

cb(null, json)

} catch(e) {

cb(-1,null)

}

} else {

cb(-1,null);

}

} else {

console.log("Request error", err)

cb(err, null)

}

})

A publisher will be able to determine the Application id and user email from the Authentication header information.

### Bewit Usage

Product publishers may want to protect access to resources to capture user consumption of such resources for audit/metrics purpose. When responding to an OpenSearch query, the publisher may want to add an additional parameter (bewit) to any provided url links. This parameter contains the necessary credentials to authenticate the request if the consumer were to decide to follow the provided link ([see](https://github.com/hueniverse/hawk#usage-example)).

## OpenSearch

Publishers and Consumers interface using [OpenSearch](http://www.opensearch.org/Home) query interface and a published endpoint using HTTP(s). The publisher may provide an OpenSearch description document to facilitate the discovery process.

### OpenSearch URL template syntax

Example: http://example.com/search?q={ProductSearchTerm}

Search parameter:

q

Available Products:

surface\_water,

daily\_precipitation,

daily\_precipitation\_24hr\_forecast,

flood\_forecast,

landslide\_forecast

Optional parameters:

startIndex

itemsPerPage

language

latitude, longitude or bbox

startTime

endTime

limit

Output Encoding

Publisher shall respond in JSON format

Output Format:

JSON Activity stream/action

<http://tools.ietf.org/html/draft-snell-activitystreams-09>

<http://tools.ietf.org/html/draft-snell-activitystreams-actions-06>

## Publish A Product

Assuming a root path of /, products could be served from /products/:type/:id as an example.

Clients need to have access to a product page for each available product.

That page could be rendered in html or JSON format based on headers in the request or url extension

Example:

<http://www.example.com/products/surface_water/193>

<http://www.example.com/products/surface_water/193.html>

<http://www.example.com/products/surface_water/193.json>

The language will be detected in the headers (from browser preferences or set by consumer HTTP Accept-Language header)

### Meta-tags

The publisher will add metatags to allow the publishing of the product as Twitter Card and Facebook OpenGraph

<https://dev.twitter.com/cards>

<https://developers.facebook.com/docs/opengraph>

<http://davidwalsh.name/facebook-meta-tags>