Turn Your Facebook App Into A List Building Machine!

Introduction.

Since releasing my various Facebook app building tools there is one question I get asked a lot.

“Can I make this gather email addresses for my opt-in list?”

To those people I have always given them a word of warning about such practices and then pointed them in the general direction of a freelance site where they could find someone at a relatively low cost who could make the necessary PHP code changes.

But since this is such a frequently asked question I decided to write a report that will show you how to modify the code yourself and get your application to add users to a mailing list.

You can use these techniques with almost any Facebook app code including that which is generated by Instant Facebook Games and Social App Creator.

For the sake of example I shall be using the template code that was provided with “Facebook Apps Made Easy”.

But before we get into how to do this I need to pass on the same word of warning that I have given to those who have asked about this.

Warning!

The more permissions you ask a Facebook user for when they first run your application, the less likely they are to hit the confirmation button. This applies in a big way to the request for email address access.

If your application is something cool and/or unique then people will almost certainly go for it, but many will be reluctant to grant such permission just for a go on your Pac-man or Tetris game.

Requesting the Users email address through Facebook

This is the easy part of the process as to get the email address we simply need to request it when Facebook asks the user for permissions (when they run your app for the first time).

You need to search inside your Facebook apps code for where the permissions are requested. The best way is to search the open PHP script for ***'scope'***

If you search the index.php file of the template code provided with "Facebook Apps Made Easy" you will find a section of code that looks like this;

```
if($me) {}
else {

        $loginUrl = $fb->getLoginUrl(array(
     'scope' => ''
        ));
                echo "
                <script type='text/javascript'>
                window.top.location.href = '$loginUrl';
                </script>
        ";
        exit;
}
```

I have highlighted the line we are interested in in red.

If the code you are modifying is written using the old Facebook development system (SDK 2.x) then the word scope may be replaced with 'req_perms'. If this is the case you will need to get a more up to date version of the code you are editing as anything not using the version 3 SDK won't work after October 2011.

You need to change the line to read:

```
'scope' => 'email',
```

More than one permission may be requested and in that case they are separated by commas. If this is the case in a script you are changing then you will have to add email to the list.

For example if changing a line that reads;

```
'scope' => ‘user_birthday’,
```

It would become;

```
'scope' => ‘user_birthday,email’,
```

A full list of all the permissions possible can be found here: http://developers.facebook.com/docs/authentication/permissions/

But the full list isn’t really of interest to us, we just want to make sure we ask for the email permission and then we can do something with that email address.

Now the next thing we need to do is to add the email address to our mailing list. But we cannot do it at the point in the code that the permission is requested as the email has not been authenticated by Facebook at this point.

We need to add the email to the mailing list if this is the first time the app has been run AFTER the permission has been granted. But in order to do this we need to identify that this is the first run.

Thankfully there is a little bit of code that redirects the user correctly after permissions are set and by making a small modification to that we can make use of it later in the script.

Look near the top of the php script and you will see a piece of code that looks like this;

```
if (isset($_GET['code'])){
    header("Location: " . $canvasPage);
  exit;
}
```

For those that care about what this is I will explain it, but as with all of this you don’t need to understand it, you just need to change it!

In a nutshell, Facebook sets a special variable called ‘code’ when the app is run for the first time directly after the permission request, and this is used so that the user can be directed correctly to the Facebook app canvas page and not the page directly on your server.

By making an easy change to this as follows (change highlighted in **bold red**);

```
if (isset($_GET['code'])){
   header("Location: " . $canvasPage.'?ft=1');
   exit;
}
```

Because this code is only ever called the first time we are making it redirect but sending a variable called ft with the value of 1. We can now detect this later in the script AFTER Facebook has passed us the users information.

The final change is the biggest of them all and the easiest way to do this is to copy and paste from one of the sample files supplied with this report and fill in your details.

The last change is the piece of code that takes the users email address and adds it to you list and depending on the list service you use there are two different methods available to you.

Method 1: If your list handles subscribe by email

Most of the major mailing list services have the facility to accept subscriptions via email. That is to say if someone sends an email (usually blank but it may have to contain the word 'join' or 'subscribe') to a special email address assigned to that list they will be added to the list.

I personally use AWeber for my lists as they have the best deliverability and features in the industry. Sure they're not the cheapest but when it comes to your mailing list you get what you pay for and you don't want to cut corners.

AWeber are one of the providers who offer subscribe by email, but in my experience it doesn't always work "out of the box." The best thing to do is to test it by sending a blank email to *yourlistname*@aweber.com and see if you get a confirmation of a bounce. If you get a bounce an email to their customer services gets things sorted quickly. Doing this test will save you a lot of potential head-scratching later – I speak from experience!

So assuming your service can handle subscription by email you will want to open the subscribebyemail.php file that was supplied with this report and copy the code from in that into your index.php file directly under the line that reads: include 'spinc.php';

In the template files this is around line 49.

When done you will have a chunk of code like this;

```
include 'spinc.php';

if (isset($_GET['ft'])){
        // Add email to mailing list using mail method.

        $useremail=$me['email'];
        $userfullname=$me['name'];

        $mail_header = 'From: '.$userfullname.' <'.$useremail.'> '."\r\n".
   'Reply-To: '.$useremail."\r\n".
   'X-Mailer: PHP/' . phpversion();

   $subject = "subscribe";
   $body = "subscribe";
   mail('listname@mailservice.com', $subject, $body, $mail_header);
        // End of optin code.
}
```

The only things you need to change are highlighted in red. The subject and body variables contain the text for the subject line and the body of the email that will be sent to the list service email. In many cases these will be blank, in which case they would look like this;

```
$subject = "";
$body = "";
```

But some providers require the word ‘join’ or ‘subscribe’ in either the subject or the body. Most don’t care in which case you can leave these alone.

The text highlighted in ***bold red italics*** is where you put your list mailing address.

That is it!

Method 2: When there is no 'subscribe my email' option.

This is a little more involved but once again there is a sample provided in the file called subscribecurl.php

This method does require that your host has curl installed. Most do, but if this doesn't work for you that would be one of the first things to check.

For this method you are going to have to work out a little bit from your HTML subscribe form, but don't worry, it's not as daunting as it sounds.

Step 1: Get the html source of your mailing list subscription form. Ensure it's the form for the correct mailing list – as most scripts put some kind of long 'form id' within the sign-up form. This code tells the list which list the form is making the request.

Make sure you get the HTML version of your form and not the javascript version.

Copy and paste the HTML code into a Notepad window on your computer.

I suggest that you copy and paste out all the relevant parts of the html code into a separate notepad document for ease.

Here are the things you need to document;

1. The post action url. There should only be one of these in your html code, I am giving a couple of examples below:

```
<form method="post" action=http://maillist.com/h/subs.php?listID=1343 >

<form method="post" class="form-wrapper" action="http://www.alisty7.com/scripts/add.php" >
```

We are only really interested in the url this points to. In the two examples above the url is highlighted in red.

2. The Name and Email name attributes. These are usually quite apparent based on the words that appear in quotes after 'name='. It is these values that you need to note down and be sure to note them exactly as they appear. In the example below the name field is called “FirstName” with capital letters on the word First and Name.

```
<input  value="" name="FirstName" type="text" size="30" >

<input  value="" name="Email" type="text" size="30" >
```

Your code will look different but the principle is always the same, all these forms will have a name and email attribute. As before I have highlighted them in red in the example above.

3. Identify any Hidden form fields and their values. You now need to identify any ‘hidden’ input fields in your form and note the ‘name’ attribute and also the value. There may be several of these. Here are some examples in which we have highlighted the name attribute in red and the value in green.

```
<input type="hidden" name="SpecialFormCode" value="9a2bba063680875730">
<input type="hidden" name="meta_web_form_id" value="1211542425" />
<input type="hidden" name="meta_split_id" value="" />
<input type="hidden" name="listname" value="magicweightloss" />
```

You will notice that one of our examples has nothing for the value, in this case you still need to keep a note of it as it is important to pass across all the hidden codes even if they have no value.

I like to document all the fields in a notepad document like this:

```
url: 'http://www.alistprovider.com/addsubscriber.php'

Form Fields:
Email
FirstName
SpecialFormCode                9a2bba063680875730
meta_split_id
listname                       mylistname
```

Now we have to transfer that information into a CURL function. If you look at the sample code provided there is a file called subscribebycurl.php which contains an example function that you can edit and fill in the gaps.

If you are using our template application from Easy Facebook Apps then this code gets inserted around line 50.

Here is a copy of the code with the changes made to reflect the example above;

```
if (isset($_GET['ft'])){
        // Add email to mailing list using CURL method.

        $useremail=$me['email'];
        $userfullname=$me['name'];
        $userfirstname=$me['first_name'];
        $curlurl='http://www.alistprovider.com/addsubscriber.php';
        // set URL and other appropriate options - use the url for your subscribe script
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, $curlurl);

        // Set up the post using all the fields you found.  At the end of each goes a comma,
except the last in the list which should
        // finish with closing the bracket and a semi colon.
        // Put fixed information in single quotes, but for fields like name and email you can
use the variables from the script
        // as per the example.
        $data = array('name' => $userfullname,
                              'email' => $useremail,
                              'meta_split_id' => '',
                              'SpecialFormCode' => '9a2bba063680875730',
                              'listname' => 'mylistname');

        // Set some curl options curl_setopt($ch,
        CURLOPT_POST, 1); curl_setopt($ch,
        CURLOPT_RETURNTRANSFER,1); curl_setopt($ch,
        CURLOPT_POSTFIELDS, $data);
        // execute cURL command:
        $exec=curl_exec($ch);

        // End of CURL Opt-in code.

}
```

The parts edited have been highlighted in red.

The most complicated part of this process is dealing with the changes in the $data array so I am going to expand on that and set out the ‘rules’ for this piece of code.

It is also worth re-iterating here – if your provider offers subscribe by email then use that instead. It is easier and more direct code.

Explaining the $data array.

This is not meant to be a technical explanation, if you already know some PHP then the following explanation isn't for you!

```
$data = array('name' => $userfullname,
                        'email' => $useremail,
                        'meta_split_id' => '',
                        'SpecialFormCode' => '9a2bba063680875730',
                        'listname' => 'mylistname');
```

This is basically where we define all the settings we documented earlier by looking at the form HTML code.

It might help you to see that this actually reads like this to the php language;

```
$data = array('name' => $userfullname, 'email' => $useremail, 'meta_split_id' => '', 'SpecialFormCode' => '9a2bba063680875730', 'listname' => 'mylistname');
```

In PHP new lines don't mean anything, it only assumes the end of a line when it sees the semi colon (;)

So really all this is made up of is each form items name in single quotes, followed by => and then the value in single quotes (no quotes if its one of the available string variables that start with the dollar symbol).

Each form item is separated by a comma. No comma is necessary after the last one, just as if you were typing up a list in a document with commas.

Important Notes about this Report

This report is being sold at a very low-cost. As such it does not include any form of technical support or assistance with php coding so please do not email questions about this as those emails will not get a response.

All the code in here and in the supplied samples has been tested with a number of mailing list providers and THE CODE WORKS.

If you experience trouble then it has to be because;

1) You have done something wrong.
2) Your list provider is acting strangely.

If you do need assistance there are some excellent php coders on gig sites like fiverr.com who will help with basic stuff like this very cheaply.