PHP AdWords API Lib Quick Start

1. What is PHP AdWords API?

PHP AdWords API Lib is middle-tier library implementing the access to the Google AdWords API from your PHP products and applications. With it's simple to use object-orientated design it helps you use Google AdWords API with outstanding ease, without the hassle of investigating, designing, coding, debugging and testing your own low-level communication layer.

2. What do I need to know first?

To use PHP AdWords API Lib you need to have installed and working PHP >= 4.3.0 and PEAR SOAP or NuSOAP toolkits. To use AdWords API Lib with PEAR you need to have xml and openssl php extensions. To use with NuSOAP you need to have curl or openssl php extensions.
You can use servercheck.php file in the root AdWords API Lib directory to check your system configuration.

3. Which SOAP library to use? PEAR SOAP or NuSOAP?

PHP AdWords API Lib works with NuSOAP as well as with PEAR SOAP. You can switch between NuSOAP and PEAR without changing any code, but you can not use both of them in a same script! There is incompatibility between them.

4. How to install PEAR SOAP or NuSOAP?

The library has NuSOAP included in the distribution. Our version is patched to work correctly with the Google AdWords API. You can also download NuSOAP. Unzip the arhive file in a directory where your scripts can find it and include nusoap.php file. You can download latest version from http://sourceforge.net/projects/nusoap/

If you have a recent PHP installation you already have PEAR. If you don't have it you can download latest PEAR package from http://pear.php.net . How to install packages from PEAR you can read in the PEAR user's manual
http://pear.php.net/manual/en/installation.php

5. How to install PHP AdWords API Lib?

Just unzip arhive file with PHP AdWords API Lib in a directory where your scripts can find it. To have the functions and classes available to your applications include the services class you want to use.

6. Which files do I need to include?

For example if you want to use Google AdGroup Service you need to include AdWordAdGroupService.inc file.

7. OK I'm ready. Give me some examples.

Let start with examples given in Google AdWords API site:
http://www.google.com/apis/adwords/php.html

First of them is “Estimating traffic for a keyword”
Let make the same example step by step with PHP AdWords API Lib using NuSOAP.
a) First step is to include all needed classes. As we are using NuSOAP we need to include nusoap.php. We also need AdWordTrafficEstimatorService class, which is used for traffic estimator service.

```
<?php
require_once('nusoap.php');
require_once('AdwordsLib/services/AdWordTrafficEstimatorService.inc');
```

b) We are ready to create an object of class AdWordTrafficEstimatorService. We need to pass the SOAP toolkit that will be used in the constructor.

```
$obj = new AdWordTrafficEstimatorService(ADWORD_SOAP_TOOLKIT_NUSOAP);
```

It is a good idea to use defined constants for supported by AdWords Lib toolkit names, which are: ADWORD_SOAP_TOOLKIT_NUSOAP – for NuSOAP, ADWORD_SOAP_TOOLKIT_PEAR – for PEAR SOAP.

Now we need to set up our new object to use our AdWords API account, by setting email, password, user agent and token.

```
$obj->setAccount($email, $pass, $useragent, $token);
```

d) Our traffic service object is ready to be used, but first we need two

AdWordKeywordRequest objects. Here is the first one:

```
$kwReq1 = new AdWordKeywordRequest();
$kwReq1->set('text', 'flowers');
$kwReq1->set('maxCpc', '50000');
$kwReq1->set('type', 'Broad');
```

As you can see, first we create an empty AdWordKeywordRequest object and then added the fields one by one. If you want you can do the same thing on one line by passing an array to the class constructor. Let's create the other one in this manner.

```
$kwReq2 = new AdWordKeywordRequest(array('text' => 'chocolates', 'maxCpc' => '50000',
'type' => 'Broad'));
```

e) That's all.We are ready to connect Google.

```
$response = $obj->estimateKeywordList(array($kwReq1, $kwReq2 ));
```

f) If something is not ok estimateKeywordList() function will return false.
All services functions return false on error! We need to check the Google's response. If there is an error lets show it and stop the script.

```
if ($response === false) {
      $error = $obj->getLastError();
      echo $error->toString();
      exit;
}
```

g) If we got this far, $response contains two objects of class AdWordKeywordEstimate, let show them to the user.

```
foreach ($response as $KeywordEstimate)  {
     $fields = $KeywordEstimate->getRow();
     foreach ($fields as $name => $value) {
          echo "$name = $value <br>";
    }
   echo '<br>';
}
?>
```

8. That was easy! But how to create a campaign with an AdGroup, Creative and Keywords.

Ok. Let's do that using PHP AdWords API Lib. The example is the same as “Creating a new campaign with an Ad Group, creative, and keywords" from the examples given in Google AdWords API site:

http://www.google.com/apis/adwords/php.html

a) This time we will use PEAR SOAP. We also need to include all needed classes to work with Campaign, AdGroup, Creative and Keyword Services.

```
<?php
require_once('nusoap.php');
require_once('AdwordsLib/services/AdWordCampaignService.inc');
require_once('AdwordsLib/services/AdWordAdGroupService.inc');
require_once('AdwordsLib/services/AdWordCreativeService.inc');
require_once('AdwordsLib/services/AdWordKeywordService.inc');
```

b) Now we need to create objects of services classes and to set them up.

```
$campServ = new AdWordCampaignService(ADWORD_SOAP_TOOLKIT_NUSOAP);
$campServ->setAccount($email, $pass, $useragent, $token);
$adgrpServ = new AdWordAdGroupService(ADWORD_SOAP_TOOLKIT_NUSOAP);
$adgrpServ->setAccount($email, $pass, $useragent, $token);
$creatServ = new AdWordCreativeService(ADWORD_SOAP_TOOLKIT_NUSOAP);
$creatServ->setAccount($email, $pass, $useragent, $token);
$keywrdServ = new AdWordKeywordService(ADWORD_SOAP_TOOLKIT_NUSOAP);
$keywrdServ->setAccount($email, $pass, $useragent, $token);
```

c) Before adding adgroups and keywords we need to create a Campaign.

```
$arr = array('name' => 'campaign one', 'dailyBudget' => '2000000');
$newCamp = new AdWordCampaign($arr);
```

d) Now we need to add some geoTargeting and languageTargeting to our Campaign class as it is done in the Google's example.

```
$arr = array('countries' => array('FR', 'GB'));
$geoTarg = new AdWordGeoTarget($arr);
$arr = array('en', 'fr');
$langTarg = new AdWordLanguageTarget($arr);
$newCamp->set('geoTargeting', $geoTarg);
$newCamp->set('languageTargeting', $langTarg);
```

e) We create two more classes to create a single Campaign but this is the long way to do this. You don't need to create classes of AdWordGeoTarget and AdWordLanguageTarget. Just pass geoTargeting and languageTargeting info as parameters to AdWordCampaign constructor and it will create this classes instead of you. Let do both points c) and d) in a short way. So instead of 9, we can use just 2 lines of code.

```
$arr = array('name' => 'campaign one', 'dailyBudget' => '2000000', 'geoTargeting' => array
('countries' => array('FR', 'GB')), 'languageTargeting' => array('en', 'fr'));
$newCamp = new AdWordCampaign($arr);
```

f) We are ready to connect Google to create the new Campaign.

```
$response = $campServ->addCampaign($newCamp);
if  ($response === false) { //check for errors
      $error = $obj->getLastError();
      echo $error->toString();
      exit;
}
//grab the ID of the new Campaign
$campID = $response->get('id');
```

g) Now we have the ID of the new campaign. Lets add an AdGroup to it.

```
$newAdgrp = new AdWordAdGroup(array('campaignId' => $campID, 'maxCpc'=> '50000'));
$response = $adgrpServ->addAdGroup($campID, $newAdgrp );
if  ($response === false) { //check for errors
      $error = $obj->getLastError();
      echo $error->toString();
      exit;
}
//grab the ID of the new AdGroup
$adgrpID = $response->get('id');
```

h) We have an adgroup, let's add some keywords to it

```
$arr =array('adGroupId' => $adgrpID, 'type' => 'Broad');
$newKeyW1 = $newKeyW2 = $newKeyW3 = new AdWordKeyword($arr);
$newKeyW1->set('text', 'springers');
$newKeyW2->set('text', 'spaniels');
$newKeyW3->set('text', 'english springer spaniel');
$response = $keywrdServ->addKeywordList($adgrpID , array($newKeyW1, $newKeyW2,
$newKeyW3));
if  ($response === false) { //check for errors
```

```
    $error = $obj->getLastError();
    echo $error->toString();
    exit;
}
```

i) If we get this far, the keywords were created and we are ready to create a creative.

```
$arr = array('adGroupId' => $adgrpID,
             'headline' => 'Rescue Springer Spaniels',
             'description1' => 'Save a Springer from the shelter.',
             'description2' => 'Get a friend for life.',
             'displayUrl' => 'www.saveSpringerSpaniels.dog',
             'destinationUrl' => 'http://www.saveSpringerSpaneils.dog');
$newCreat = new AdWordCreative($arr);
$response = $creatServ ->addCreative($newCreat);
if ($response === false) { //check for errors
    $error = $obj->getLastError();
    echo $error->toString();
    exit;
}
?>
```

That's all. We have a working campaign with adwords, creative and keywords.

9. Tips

a) In Google AdWords you can edit a client's account rather than your own account. To call estimate traffic for the keywords from the previous example, but for a different client that is managed by you, just pass client's email to the function as last parameter:

```
$response = $obj->estimateKeywordList(array($kwReq1, $kwReq2 ), $clientEmail);
```

You can use as many client's accounts as you want with one service class. If you call many functions for one client's account it's better to set that account to the service.

```
$obj = new AdWordTrafficEstimatorService(ADWORD_SOAP_TOOLKIT_NUSOAP);
$obj->setAccount($email, $pass, $useragent, $token);
...
$obj->setClientEmail($clientEmail);
$response = $obj->estimateKeywordList(array($kwReq1, $kwReq2 ));
```

b) Use one service class per Google AdWords service.

```
$obj1 = new AdWordTrafficEstimatorService(ADWORD_SOAP_TOOLKIT_NUSOAP);
```

```
$obj1->setAccount($email, $pass, $useragent, $token);
$obj2 = new AdWordTrafficEstimatorService(ADWORD_SOAP_TOOLKIT_NUSOAP);
$obj2->setAccount($email2, $pass2, $useragent2, $token2);
...
$response1 = $obj1->estimateKeywordList(array($kwReq1, $kwReq2 ));
$response2 = $obj2->estimateKeywordList(array($kwReq1, $kwReq2 ));
```

This code will work two times slower than this one:

```
$obj = new AdWordTrafficEstimatorService(ADWORD_SOAP_TOOLKIT_NUSOAP);
$obj->setAccount($email, $pass, $useragent, $token);
...
$response1 = $obj->estimateKeywordList(array($kwReq1, $kwReq2));
$obj->setAccount($email2, $pass2, $useragent2, $token2);
$response2 = $obj->estimateKeywordList(array($kwReq1, $kwReq2));
```

c) You can create a campaign this way

```
$arr = array('name' => 'campaign one', 'dailyBudget' => '2000000');
$newCamp = new AdWordCampaign($arr);
$arr = array('countries' => array('FR', 'GB'));
$geoTarg = new AdWordGeoTarget($arr);
$arr = array('en', 'fr');
$langTarg = new AdWordLanguageTarget($arr);
$newCamp->set('geoTargeting', $geoTarg);
$newCamp->set('languageTargeting', $langTarg);
```

If you just pass the arrays for AdWordGeoTarget and AdWordLanguageTarget constructors as parameters to AdWordCampaign constructor, it will create these classes instead of you.

```
$arr = array('name' => 'campaign one', 'dailyBudget' => '2000000', 'geoTargeting' => array
('countries' => array('FR', 'GB')), 'languageTargeting' => array('en', 'fr'));
$newCamp = new AdWordCampaign($arr);
```