OAuth

For WHMCS

A basic guide for functionality and usage

By: WHMCS Addon

Contents

Introduction

This addon takes the WHMCS API to a whole other level affectively reducing the limitation of the existing WHMCS API. Using OAuth removes the need to grant specific admin's API access which could be potential security flaw. As well using OAuth there is no longer a need to restrict IP access to the API.

OAuth secures the API by using a secret key and public key system. The secret key is known between the application and the server only and is encoded with the public key. To ensure the validity of the request a timestamp and nonce is also encoded with the private key and public key.

The timestamp makes any request time sensitive. While the nonce prevents a request from being run multiple times on a server (replay attack).

After which the request can be made by sending OAuth specific data. This includes the public key, encoded key, timestamp, and nonce. An example of the request can be found later in this document.

Setup

Setting up the addon is simple. Extract the files into your WHMCS root directory. Copy the template file "templates/default/oauth.tpl" into your appropriate template folder.

Login to your WHMCS admin area and go to Setup -> Addon Modules. Once there press activate on the OAuth module. You should now see OAuth settings appear. You now have the option to enable OAuth and set the users who can manage the OAuth module.

If the enable button is not checked the OAuth server will not handle any requests coming to it.

OAuth Manager Tour

The OAuth manager is where you manage OAuth. From here you should see three options on the left hand side. “Create OAuth Key”, “OAuth Keys”, and “OAuth Maintenance”.

OAuth Keys

By default you are viewing the “OAuth Keys” page. This is a page where you can view all the available keys, delete keys, and edit keys.

Create OAuth Key

This is where you can create new OAuth keys. You will be prompted to name your key. Optionally you can disallow the specific key access to any API functions you want.

If you have OAuth Pro enabled you will notice extra functionality.

You may notice you have the ability to select a specific user for the key. Leaving it as admin key will keep it private for your admin applications that need access to your websites functionality. Otherwise you can assign a key to any of your clients.

Another function you may notice is that you have the ability to allow specific functions for your client to use. Your clients are subject to the master function list assigned in the OAuth Pro setting’s menu. Using the allow function option you can allow your client to use a specific function that you do not want other clients to have access to.

OAuth Maintenance

This page offers two specific functions.

Empty Logs: OAuth stores logs of every request made to the server. Every once in a while you may want to clear these logs to reduce the database size and make the OAuth server run a bit faster.

Uninstall: Deactivating the addon through your Addon Manager will not uninstall it and remove the existing keys. Pressing this button completely remove all the OAuth records from your database.

Settings

If you have activated OAuth Pro then this page is available for you.

On this page you should see two options “Allow Clients” and “Client Functions”. Allow clients lets you turn on and off the OAuth support for clients. Client Functions is a master list of what functions are available to your clients. By default OAuth Pro does not enable any functions. This is because certain functions can cause potential security flaws in your website. Be careful what functions you enable as you may accidentally grant access for your client to respond to support tickets, delete users, and so on…

What functions should you grant access to your client?

This is a very complicated issue. It really depends on what you are trying to accomplish with your API. Most likely you will need to visit the API market place or get a custom API developed. Before enabling any functionality for your clients please consult the WHMCS API documentation.

Please note any API functions that require a client id will attempt to override the inputted client id with the requesting user's client id. This may not work on all servers and should be tested before assuming this action is working. That being said this will not have any effect on API's that uses client email addresses (some API's from WHMCS take either a client id or client email address).

API Market Place

We are constantly working to release API's in the API market place both from us (WHMCS Addon) and other developers throughout the WHMCS community. With time we hope to be able to cover the majority of major requests for more API functionality.

API's maybe offered for free or at additional cost at the discretion of the API developer. We try our best to ensure the developers release a stable API before it is released into the market place. By no means do 3rd party API's represent WHMCS Addon brand.

Using OAuth

The following is an example of how to make an OAuth call.

```
<?php

$oauth_secret = "7504825008f9ab99682e3e71d60423d4154bfcc9";

// Define the data being posted to the server

$postfields = array(

      'oauth_consumer_key' =>
"cb811ca790374ebcd6484c5414184d76c4eadae9",

      'oauth_nonce' => uniqid('', true),

      'oauth_timestamp' => time(),

      'oauth_version' => "1.0",

      'action' => 'getclientsdetails',

      'clientid' => '1',

      );

// Create the OAuth token

$postfields["oauth_token"] =
sha1(base64_encode($postfields["oauth_nonce"].$postfields["oauth_time
stamp"].$postfields["oauth_consumer_key"].$oauth_secret));

// CURL code to make request

$ch =
curl_init("https://example.com/modules/addons/oauth/oauth_server.php"
);

curl_setopt($ch, CURLOPT_HEADER, 0);

curl_setopt($ch, CURLOPT_POST, 1);

curl_setopt($ch, CURLOPT_POSTFIELDS, $postfields);

curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

$output = curl_exec($ch);

curl_close($ch);

// Decodes output into PHP variables

$json = json_decode($output, true);
```

Breaking down this example you will notice a few variables.

The $oauth_secret variable holds the OAuth secret key. In this case the secret key is "7504825008f9ab99682e3e71d60423d4154bfcc9"

```
$oauth_secret = "7504825008f9ab99682e3e71d60423d4154bfcc9";
```

The postfields array handles what will be sent to the OAuth server.

- oauth_consumer refers to the public key you are using.
- oauth_nonce is just a bunch of random characters generated by PHP. The uniqid() will make sure that these random characters are never the same for any request.
- oauth_timestamp holds the current time. Since it generates a timestamp this should be the same time on any server. No need to change the timezone.
- oauth_version simply holds the version number for OAuth to use. This can only be set to "1.0" at the moment.
- action tells the system what API to use. In this case we are using "getclientsdetails"
- clientid is simply the client's id we are getting info for. It is a requirement of this specific API request.

Once all those variables are set we create the OAuth token. This is an encoded key that holds a bunch of the OAuth data. It needs to be encoded in this exact order to be recreated on the server side. Otherwise the request will fail from an invalid token.

```
$postfields["oauth_token"] =
sha1(base64_encode($postfields["oauth_nonce"].$postfields["oauth_timestamp"].
$postfields["oauth_consumer_key"].$oauth_secret));
```

Next we setup the CURL data. In this case we are connecting to the WHMCS installation installed at example.com .

```
$ch = curl_init("https://example.com/modules/addons/oauth/oauth_server.php");
```

After we make the CURL call, we take the result and decode it using the following method:

```
$json = json_decode($output, true);
```

The results are now held in the array $json .

HTACCESS Tricks

If you have not noticed the location "modules/addons/oauth/oauth_server.php" is not a pretty url to go to, especially if you are distributing this API to your clients. A simple fix to this would be adding a line like this into your HTACCESS file:

```
RewriteRule ^api$ modules/addons/oauth/oauth_server.php [L,NC]
```

This will make the examples CURL request to: `https://example.com/api`