Setting up your Cron

> This page concerns phpList self-hosted users only. If you have a registered account at the phpList Hosted service, please contact hosted@phpList.com.

What is cron

Cron is a time based task scheduler on Unix-like operating systems. For more information check the Wikipedia page. This chapter will describe how to set this up on Linux. On other Unix systems the set up will be similar.

Why set up cron

When you set up the cron, two of the most time consuming parts of phpList will become automated. These are processing the queue and processing the bounces. Once you set up the cron, you will not have to do these in your browser, but they will happen automatically. Sending a campaign will become as easy as sending a normal email from your desktop.

Caveats

The method described in this chapter requires the CLI version of PHP. This may not be available on all systems. There are some "hacks" to make it work with other versions of PHP, but this is non-intentional functionality and therefore not described here.

Step 1: set up commandline

The easiest way to set up the cron is to set up a commandline script that will handle all phpList commands. There is an example in the "bin" directory in the phpList archive file.

Let's call the commandline script "phplist" and edit it with your favourite text editor:

```
$ nano phplist
```

and type (or copy-paste) the content (note the line breaks; the following command should be pasted literally as two lines):

```
#! /bin/bash
/usr/bin/php /home/website/public_html/lists/admin/index.php -c
/home/website/public_html/lists/config/config.php $*
```

You will need to change the above content to fit your system:

#!/bin/bash -> the shell that you want to use

/usr/bin/php -> the path to the PHP-cli commandline interpreter, this can vary per system. Debian based systems (including Ubuntu) should install the "php-cli" package.

/home/website/public_html/lists/admin/index.php -> the path to your phpList installation, pointing to the index.php file in the admin directory of phpList

/home/website/public_html/lists/config/config.php -> the path to your config file

Once you have constructed your commandline file, place it somewhere in your path, e.g. in /usr/local/bin and make it executable.

```
chmod 755 /usr/local/bin/phplist
```

From then on, you can process phpList on commandline, for example with

```
phplist -pprocessqueue
```

Step 2: set up the crons

Once you have the commandline script, you can set up the crons with your favourite cron editor. On Linux, you type "crontab -e" in order to edit your cron entries.

```
0-59/5 * * * * phplist -pprocessqueue > /dev/null 2>&1
0 3 * * * phplist -pprocessbounces > /dev/null 2>&1
```

The above example will process the queue once every 5 minutes and process the bounces once a day. That will be sufficient for most systems.

The example also discards any output, which you will want to do. You can always check the output by running the above commands manually from the shell-prompt without

```
> /dev/null 2>&1
```

On CentOS 7 you need to specify the full path to the bash script. Your cron entries should look like this:

```
0-59/5 * * * * /usr/local/bin/phplist -pprocessqueue > /dev/null 2>&1
0 3 * * * /usr/local/bin/phplist -pprocessbounces > /dev/null 2>&1
```

Step 3: tell phpList

Once you've set up the cron and it is working, you can tell phpList and some links will disappear. Put the following two lines in your config file:

```
define ("MANUALLY_PROCESS_BOUNCES",0);
define ("MANUALLY_PROCESS_QUEUE",0);
```

This will hide the links to process the queue and bounces from the phpList interface.

Remote queue processing

The remote queue processing is often called “phpList cron job” due to its similarity of other automatic cron jobs. Before version 3.1 there was not a standard option to process the queue from a different location than the same server your phpList instance is installed. From version 3.1.1 onwards there is an option to remotely process the queue. It is always highly recommended to use the latest phpList version for security updates and for your own convenience as latest versions offer more functionalities.

In **Config -> Settings** under the **Security** section, there will be an option called "Secret for remote processing". The first time, it will generate a random code, which you can use, or you can set your own. To run the queue remotely, all you need to do is load the following page:

http(s)://yoursite.com/lists/admin/?page=processqueue&secret=XXXX

Where the XXXX is the value that is set for the "Secret for remote processing"

Use the Hosted Service

phpList.com now has a “secret” way to get our servers to process your queue for you. There are no payment plans and if you use this, you will help us to tweak the way it works, and in return we will process your queues without charge.
We offer this, because the last thing we'd want is that the queue processing is an obstacle for using phpList. From any point of view, but mostly performance, we advise the use of commandline processing because that is the most powerful way of sending. phpList Hosted is using commandline for everything. But in some cases this may not be an option. This is where our service hopefully fulfills a need.

To use the phpList.com queue processing service sign up at https://www.phplist.com/.

Once you signed up, get your API key.
The first time you try to manually process your queue, you will get the option to set up processing with phpList.com. You can use that to set up the API key.
Alternatively go to:

```
http://www.yoursite.com/lists/admin/?page=hostedprocessqueuesetup
```

From then on, when you place a campaign in the queue, or requeue an existing one, it will activate the processing from our servers. We will run the queue very often and make sure your campaigns are sent.

If you place an embargo on a campaign, the hosted queue processing service will detect this and wait until your embargo has passed to send your campaigns.

We do keep track of the statistics, and we will give you a summary in your account pages. The phpList.com service uses the exact same system as the normal remote processing outlined above and we have no other access to your system.

Notes

There's no danger in overloading your server. When a second queue processing command is activated when the previous one is still active, phpList will detect this and bail out. Therefore there will always be one active queue sending process at any time. This is also important to ensure that the sending limits, set with Batch processing, will be honoured.

Appendix

How safe is this?

Is it not easy for someone to steal my secret? Particularly when you call the parameter “secret”.
It is quite safe, but you need to determine this for yourself. The output of the page when using the remote secret is minimised to only the statistics. If an “attacker” gets hold of your secret, all they would do is help you process your queue. In fact, if you run the queue this way yourself, you will be logged out from phpList. The access with the secret is restricted to processing the queue and nothing else. We will later on extend this to processing bounces as well. In comparison, using a “username=X&password=Y” processing URL would allow anyone snooping that data to have access to your entire phpList system.Also, if you use HTTPS, chances of someone stealing your secret are way lower.

What about timeouts, how long will the request last?

When run from a webpage, phpList, by default, will only run the queue processing for one minute. It will try to send as many mails as possible in that time. Then you will need to run it again.

Feedback

Discuss this chapter here.

Revision #4
Created 29 May 2019 09:48:07 by mariana
Updated 8 February 2022 11:40:10 by mariana