

# WebP Generator Module



version 1.0.4  
by PrestaChamps

Requirements	2
Module overview	3
Module Configuration	4
EWWW Cloud Convert	6
CWEBP Conversion Settings	7
Regenerate WebP	8
WEBP Browser Support	10
Prestashop 1.6 compatibility	11

## Requirements

Please ensure that your PHP version is at least 5.6 otherwise the module will not work (as mentioned in the product requirements sheet on Addons PrestaShop)

On the server one of the next PHP modules must be configured:

**Cwebp** => php exec function enabled

**Imagick** => Imagick php module with webp compiled

**Gmagick** => Gmagick php module with webp compiled

**Gd** => Gd php module with imagewebp, imagecreatefrompng and imagecreatefromjpeg functions

**Ewww** => ewww.io subscription and php curl service



## Module overview

This module is an essential element if you want a fast website.

The WebP Generator module, created for Prestashop sites, will allow users to efficiently regenerate their product, category and manufacturer images thereby increasing the speed of the site.

This new technology provides superior lossless and lossy compression for images on your website. After you start generating images through our module it will free up space on your server. Moreover, the quality of your images will not be affected.

The images generated through the **WebP generator module** are 26% smaller in size compared to PNGs. WebP lossy images are 25-34% smaller than comparable JPEG images at equivalent SSIM quality index.

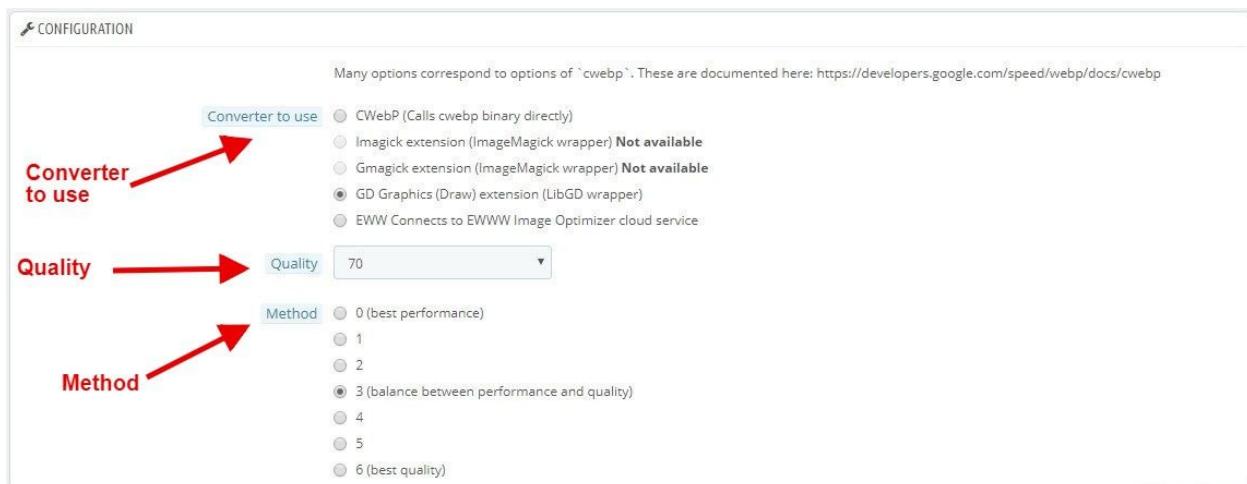
**Info:** - the module can be installed from the website back office, just like all the other modules.

## Module Configuration

- After installation the module can be accessed by clicking on the **Configure** button.



- **Converter to use** - Set the conversion methods to use depending on your server setup
- **Quality** - Specify the compression factor for RGB channels between 0 and 100
- **Method** - This parameter controls the trade off between encoding speed and the compressed file size and quality. Possible values range from 0 to 6. 0 is fastest. 6 results in best quality.

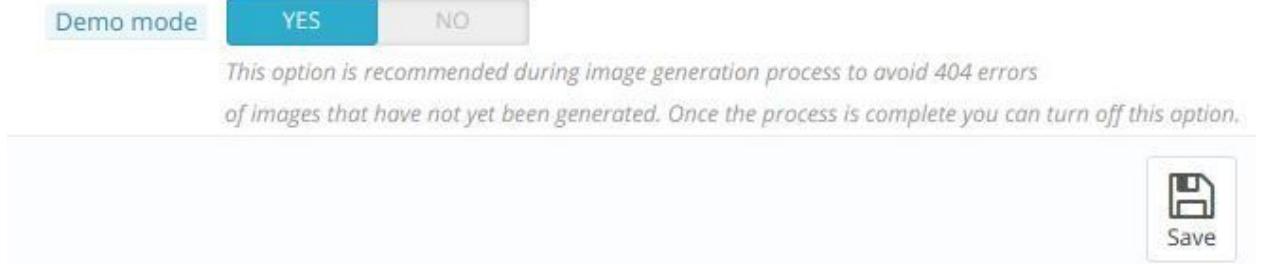


- **Low memory** - Reduce memory usage of lossy encoding at the cost of ~30% longer encoding time and marginally larger output size.
- **Lossless** - Encode the image without any loss. The option is ignored for PNG's (forced true)
- **Metadata** - Metadata to copy from the input to the output if present.

# WebP Generator Module v1.0.4



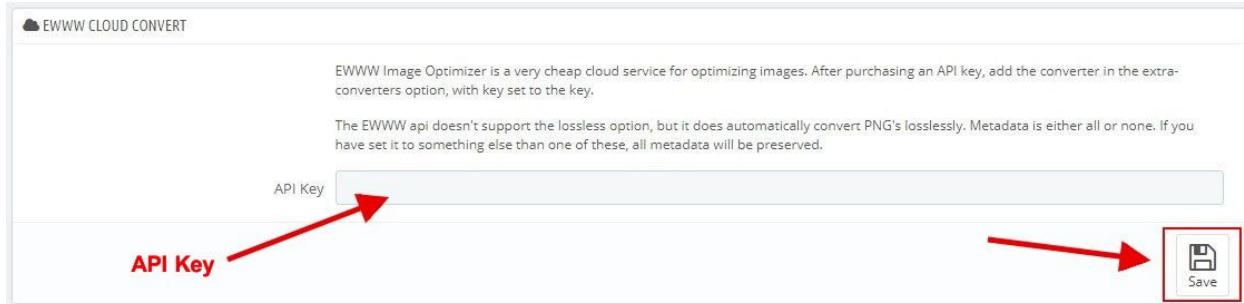
- **Demo mode** - Please enable the “**demo mode**” during the image generation process. This is recommended to avoid 404 errors of images that have not yet been generated. Once the process is complete you can disable this option.



## EWWW Cloud Convert

EWWW Image Optimizer is a very cheap cloud service for optimizing images. After purchasing an API key, add the converter in the extra-converters option, with the key field containing the API key.

The EWWW API doesn't support the lossless option but it does automatically convert PNGs losslessly. Metadata is either all or none. If you have set it to something else than one of these, all metadata will be preserved.



## CWEBP Conversion Settings

- **Use `nice` command** - If `nice` command is found on host, binary is executed with low priority in order to save system resources
- **Try common system paths** - This option tests whether cwebp is available in a common system path (eg /usr/bin/cwebp ..)
- **Try supplied binary** - If CWebP is not installed on the server, then supplied binary is selected from Converters/Binaries (according to OS) - after validating checksum
- **Turns auto-filter on** - This algorithm will spend additional time optimizing the filtering strength to reach a well-balanced quality. Unfortunately, it is extremely expensive in terms of computation. It takes about 5-10 times longer to do a conversion. A 1MB picture which perhaps typically takes about 2 seconds to convert, will take about 15 seconds to convert with auto-filter. So in most cases, you will want to leave this at its default, which is off.
- **Command line options** - This allows you to set any parameter available for cwebp in the same way as you would do when executing cwebp. You could for example set it to &quot;-sharpness 5 -mt -crop 10 10 40

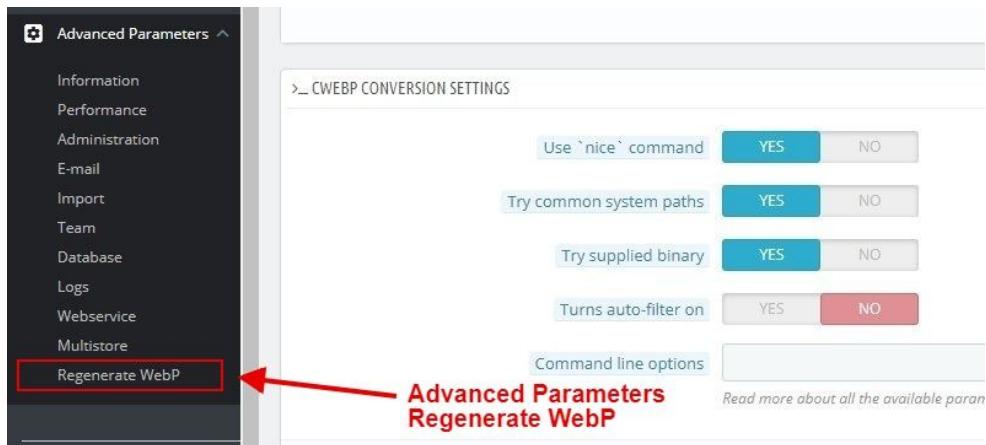


Read more about all the available parameters here:

[https://developers.google.com/speed/webp/docs/cwebp#additional\\_options](https://developers.google.com/speed/webp/docs/cwebp#additional_options)

## Regenerate WebP

After you have configured the module please go to the “**Advanced Parameters**” in the left menu. Choose the “**Regenerate WebP**” tab.



You can choose which images you want to generate

- Product Images
- Category Images
- Supplier Images
- Store Images
- Manufacturer Images

Once you've chosen, you can start generating. You can see how many images there are for each type and the processing bar will show the percent of the generated images.

The screenshot shows a module titled "REGENARATE WEBP IMAGES". A red box highlights the "PRODUCT IMAGES" section, which includes a button labeled "PRODUCT IMAGES" and a progress bar showing "0/20588". A red arrow points from the text below to this highlighted area.

Category	Count	Progress Bar
PRODUCT IMAGES	0/20588	0%
CATEGORY IMAGES	0/0	0%
SUPPLIER IMAGES	0/1	0%
STORE IMAGES	0/5	0%
MANUFACTURER IMAGES	0/6	0%

**IMPORTANT:** The window or tab where you started the regeneration process **must remain open throughout the entire process**. Closing it will interrupt the process and it will need to be reinitiated.



## WEBP Browser Support

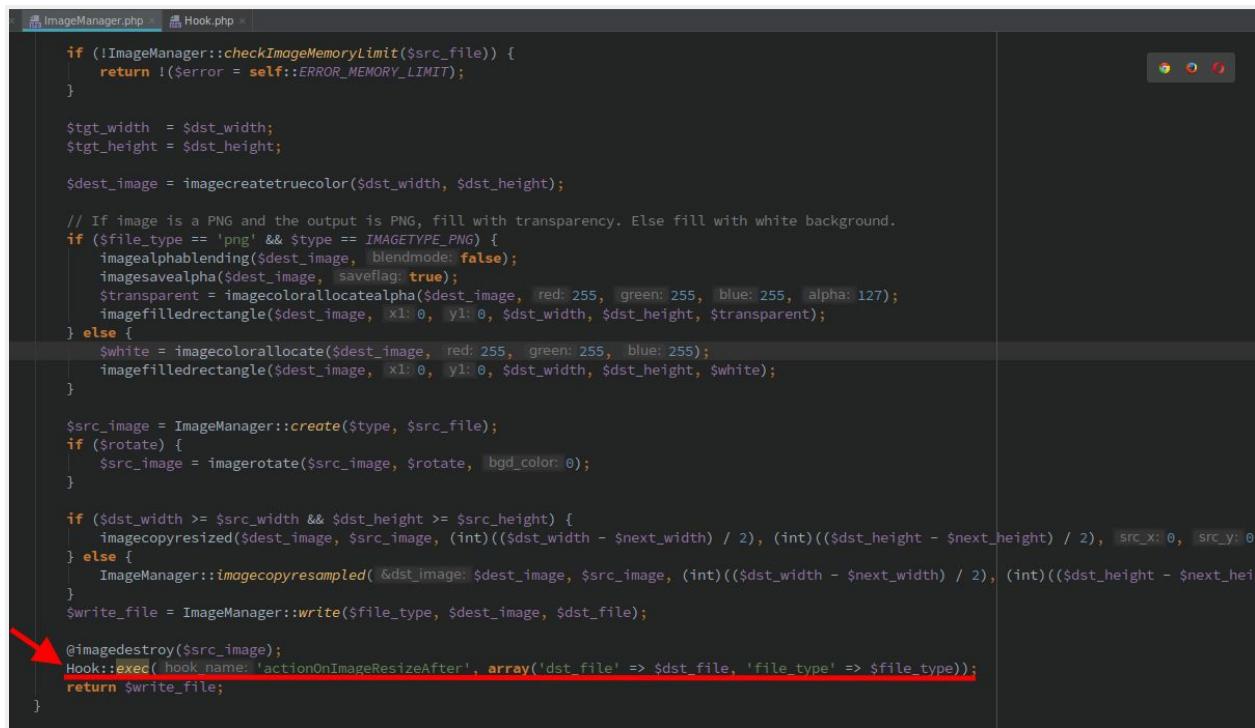
Currently only Google Chrome and Opera support WebP images. Although other browsers, such as Firefox, Safari or Internet Explorer do not currently support the image format natively, you do not have to worry. Your site's images will not be broken even for clients using these browsers, the images will appear in PNG or JPG format.



## Prestashop 1.6 compatibility

Since Prestashop 1.6 does not execute the `actionOnImageResizeAfter` hook, you need to copy the following code snippet to the `ImageManager` class `resize` method, before the `return $write_file` statement:

```
Hook::exec('actionOnImageResizeAfter', array('dst_file' => $dst_file, 'file_type' => $file_type));
```



```
if (!ImageManager::checkImageMemoryLimit($src_file)) {
    return !($error = self::ERROR_MEMORY_LIMIT);
}

$tgt_width = $dst_width;
$tgt_height = $dst_height;

$dest_image = imagecreatetruecolor($dst_width, $dst_height);

// If image is a PNG and the output is PNG, fill with transparency. Else fill with white background.
if ($file_type == 'png' && $type == IMAGETYPE_PNG) {
    imagealphablending($dest_image, blendmode: false);
    imagesavealpha($dest_image, saveflag: true);
    $transparent = imagecolorallocatealpha($dest_image, red: 255, green: 255, blue: 255, alpha: 127);
    imagefilledrectangle($dest_image, x1: 0, y1: 0, $dst_width, $dst_height, $transparent);
} else {
    $white = imagecolorallocate($dest_image, red: 255, green: 255, blue: 255);
    imagefilledrectangle($dest_image, x1: 0, y1: 0, $dst_width, $dst_height, $white);
}

$src_image = ImageManager::create($type, $src_file);
if ($rotate) {
    $src_image = imagerotate($src_image, $rotate, bgd_color: 0);
}

if ($dst_width >= $src_width && $dst_height >= $src_height) {
    imagecopyresized($dest_image, $src_image, (int)((($dst_width - $next_width) / 2), (int)((($dst_height - $next_height) / 2), src_x: 0, src_y: 0));
} else {
    ImageManager::imagecopyresampled(&$dest_image, $dest_image, $src_image, (int)((($dst_width - $next_width) / 2), (int)((($dst_height - $next_height) / 2), dst_x: 0, dst_y: 0));
}
$write_file = ImageManager::write($file_type, $dest_image, $dst_file);

@imagedestroy($src_image);
Hook::exec('actionOnImageResizeAfter', array('dst_file' => $dst_file, 'file_type' => $file_type));
return $write_file;
}
```

