GrowlHandler Book

Laurent Laville

GrowlHandler Book

Laurent Laville

Table of Contents

This complete guide documents GrowlHandler 1.0.0, published on 2015-05-11.

Part I. Getting Started

Chapter 1. Install via Composer

This handler what is not part of standard Monolog distribution, is available on Packagist bartlett/monolog-growlhandler [http://packagist.org/packages/bartlett/monolog-growlhandler] and as such installable via Composer [http://getcomposer.org/].

```
$ php composer.phar require bartlett/monolog-growlhandler
```

Chapter 2. First notifications

With this first example, we want to be notified with Growl as soon as a critical condition occurs.

Using GrowlHandler, is no more no less simple, than with other monolog handlers.

GrowlHandler class constructor requires :

- a growl configuration (`array`) or instance (`Net_Growl`) as first argument,
- the minimum logging level at which this handler will be triggered as second argument,
- and whether the messages that are handled can bubble up the stack or not as third argument.

So to receive only CRITICAL events or higher, you have just to set the logging level to `Logger::CRITICAL`.

```
<?php

try {
    $growl = new GrowlHandler(
        array(), // with all default options
        Logger::CRITICAL
    );

} catch (\Exception $e) {
    // Growl server is probably not started
    echo $e->getMessage(), PHP_EOL;
}
```

DO NOT forget to try-catch instance creation, because it will attempt to connect to server, and raise a `Net_Growl_Exception` if its impossible.

Of course it may be combined with any other monolog handler. Here is now the full script:

```
<?php
require_once 'vendor/autoload.php';

use Bartlett\Monolog\Handler\GrowlHandler;

use Monolog\Logger;

// Create the logger
$logger = new Logger('my_logger');

// Create some handlers
try {
    $growl = new GrowlHandler(
        array(), // with all default options
        Logger::CRITICAL
    );

    $logger->pushHandler($growl);

} catch (\Exception $e) {
    // Growl server is probably not started
    echo $e->getMessage(), PHP_EOL;
}

// You can now use your logger
$logger->addInfo('My logger is now ready');

$logger->addError('An error has occured.');

try {
    throw new \RuntimeException();

} catch (\Exception $e) {
    $logger->addCritical(
        'A critical condition has occured. You will be notified by growl.',
        array('exception' => $e)
    );
}
```

Chapter 3. Summary

Let’s review what we’ve done :

- installed the latest stable version using Composer.
- built your first notifications condition.

Chapter 4. Next

Choose your way depending of your skill level.

Read more

- Want to learn how to configure your Net_Growl instance, or limit notifications to certain log records, have a look on Part II, “Developer Guide”

Part II. Developer Guide

Configuration

See Chapter 5, *Configuration*

Filters

See Chapter 6, *Filters*

Chapter 5. Configuration

5.1. Some options

There are too many options to configure your Growl Application, and we will not see it all, but just few of them.

In following example, our goal is to see how to change easily, look and feel of your application, displayed by your Growl client.

We will :

- change default icon of application from `growl.png` to `growl-starkicon.png` [1]
- change default icon of each notification channel (log level for Monolog),
  - `green.png`, for debug, info
  - `yellow.png`, for notice, warning
  - `red.png`, for error, critical, alert, emergency
- change the default line formatter to include the level name about message displayed

We will also filter messages, with help of `CallbackFilterHandler` [2]

- allow notification of log records less than warning
- allow notification of message starting with, *An error has occured*, only if contextual data `exception` is present.

Filter rules.

```
<?php
$filters = array(
    function ($record) {
        if ($record['level'] < Logger::WARNING) {
            return true;
        }
        if (!array_key_exists('exception', $record['context'])) {
            return false;
        }
        return (preg_match('/^An error has occured/', $record['message']) === 1);
    }
);
```

To change notification’s icons, we have to :

- specify the resource folder `$resourceDir`, where to find images
- specify for each notification channel, what new icon will replaced the default one

  ```
  <?php
  ```

[1] http://www.iconarchive.com/show/stark-icons-by-fruityth1ng/Growl-icon.html
[2] http://php5.laurent-laville.org/callbackfilterhandler/

```
use Bartlett\Monolog\Handler\GrowlHandler;

    $resourceDir   = __DIR__ . DIRECTORY_SEPARATOR . 'images' . DIRECTORY_SEPARATOR;
    $notifications = array(
        GrowlHandler::DEBUG => array(
            'icon'    => $resourceDir . 'green.png',
        ),
        GrowlHandler::INFO => array(
            'icon'    => $resourceDir . 'green.png',
        ),
        GrowlHandler::NOTICE => array(
            'icon'    => $resourceDir . 'yellow.png',
        ),
        GrowlHandler::WARNING => array(
            'icon'    => $resourceDir . 'yellow.png',
        ),
        GrowlHandler::ERROR => array(
            'icon'    => $resourceDir . 'red.png',
        ),
        GrowlHandler::CRITICAL => array(
            'icon'    => $resourceDir . 'red.png',
        ),
        GrowlHandler::ALERT => array(
            'icon'    => $resourceDir . 'red.png',
        ),
        GrowlHandler::EMERGENCY => array(
            'icon'    => $resourceDir . 'red.png',
        ),
    );
```

To change application icon, that identified it in your Growl client :

```
<?php

    $options = array(
        'AppIcon' => $resourceDir . 'growl-starkicon.png',
    );
```

Have a look on the PEAR Net_Growl documentation [http://growl/laurent-laville.org], to learn more about all features, and options.

Now your full script should be like this one :

```
<?php
require_once 'vendor/autoload.php';

use Bartlett\Monolog\Handler\GrowlHandler;
use Bartlett\Monolog\Handler\CallbackFilterHandler;

use Monolog\Logger;
use Monolog\Handler\RotatingFileHandler;
use Monolog\Formatter\LineFormatter;

// Create the logger
$logger = new Logger('growl_conf');

// Create filter rules
$filters = array(
    function ($record) {
        if ($record['level'] < Logger::WARNING) {
            return true;
        }
        if (!array_key_exists('exception', $record['context'])) {
            return false;
        }
        return (preg_match('/^An error has occured/', $record['message']) === 1);
    }
);

// Create some handlers
$stream = new RotatingFileHandler(__DIR__ . DIRECTORY_SEPARATOR . 'growl_conf.log');
$stream->setFilenameFormat('{filename}-{date}', 'Ymd');

$logger->pushHandler($stream);

try {
    $resourceDir   = __DIR__ . DIRECTORY_SEPARATOR . 'images' . DIRECTORY_SEPARATOR;
    $notifications = array(
        GrowlHandler::DEBUG => array(
            'icon'    => $resourceDir . 'green.png',
        ),
        GrowlHandler::INFO => array(
            'icon'    => $resourceDir . 'green.png',
        ),
        GrowlHandler::NOTICE => array(
            'icon'    => $resourceDir . 'yellow.png',
        ),
        GrowlHandler::WARNING => array(
            'icon'    => $resourceDir . 'yellow.png',
        ),
        GrowlHandler::ERROR => array(
            'icon'    => $resourceDir . 'red.png',
        ),
        GrowlHandler::CRITICAL => array(
            'icon'    => $resourceDir . 'red.png',
        ),
        GrowlHandler::ALERT => array(
            'icon'    => $resourceDir . 'red.png',
        ),
        GrowlHandler::EMERGENCY => array(
            'icon'    => $resourceDir . 'red.png',
        ),
    );
    $options = array(
        'AppIcon' => $resourceDir . 'growl-starkicon.png',
        // if you have troubles with Net_Growl then debug requests to a local file
        //'debug'   => __DIR__ . '/net_growl_debug.log',
    );

    $growl = new GrowlHandler(
        array(
            'name'          => 'My Custom Growl',
            'notifications' => $notifications,
            'options'       => $options,
        )
    );
    $growl->setFormatter(
        new LineFormatter("%message%\n%level_name%")
    );

    $logger->pushHandler(new CallbackFilterHandler($growl, $filters));

} catch (\Exception $e) {
    // Growl server is probably not started
    echo $e->getMessage(), PHP_EOL, PHP_EOL;
}

// You can now use your logger
$logger->addInfo('My logger is now ready');

// This record won't be stopped by the $filters rules, but by the growl $notifications co
$logger->addDebug('A debug message.');

$logger->addError('An error has occured. Will be logged to file BUT NOT notified by Grow

try {
    throw new \RuntimeException();

} catch (\Exception $e) {
    $logger->addCritical(
        'An error has occured. Will be logged to file AND notified by Growl.',
        array('exception' => $e)
    );
}
```

5.2. Next

Read more

- Learn more about filter feature. See Chapter 6, *Filters*

Chapter 6. Filters

6.1. LogLevel Strategy

Monolog uses only the eight RFC 5424 [http://tools.ietf.org/html/rfc5424] levels (debug, info, notice, warning, error, critical, alert, emergency) for basic filtering purposes.

If it’s enough to limit log records received by each handler, we need more flexibility with GrowlHandler to avoid to be spammed with ton of log records.

We have a start of solution with the `FilterHandler` available in Monolog since version 1.8.0, that filter log records on one or more levels. But this is not without counted the new `CallbackFilterHandler` that raised this limitation, and allow advanced filtering behaviours (you can filter on each log record property).

6.2. Advanced filtering Strategy

Suppose you want to be notified at end of a long process with a specific message or contextual data, `CallbackFilterHandler` is what you’re waiting for.

It can be combined as a wrapper with any other monolog handlers, and not just the GrowlHandler.

Here we will only demonstrate its capabilities in one example. We simulate a file queue processing that log all filenames and a message when it’s over.

```
<?php
require_once 'vendor/autoload.php';

use Monolog\Logger;
use Monolog\Handler\RotatingFileHandler;
use Monolog\Processor\PsrLogMessageProcessor;

$processors = array(
    new PsrLogMessageProcessor(),
);

// Create the logger
$logger = new Logger('long_process', array(), $processors);

// Create some handlers
$stream = new RotatingFileHandler(__DIR__ . DIRECTORY_SEPARATOR . 'long_process.log');
$stream->setFilenameFormat('{filename}-{date}', 'Ymd');

$logger->pushHandler($stream);

// Processing each file in a queue
$queue = new \SplQueue();
for ($i = 1; $i < 10; $i++) {
    $queue->enqueue( sprintf('File_%02d.txt', $i) );
}
$fileCount = count($queue);

while (!$queue->isEmpty()) {
    $file = $queue->dequeue();

    $logger->addInfo('Processing file "{filename}"', array('filename' => $file));
    echo '.';

    // simulate the long process
    sleep(1);
}

$logger->addInfo(
    'Long Process with {count} files, is over !',
    array('count' => $fileCount)
);
```

At this step, if we just push an instance of GrowlHandler to the logger stack, as following

```
<?php
use Bartlett\Monolog\Handler\GrowlHandler;

try {
    $growl = new GrowlHandler(
        array(), // with all default options
        Logger::INFO
    );

    $logger->pushHandler($growl);

} catch (\Exception $e) {
    // Growl server is probably not started
    echo $e->getMessage(), PHP_EOL;
}
```

We will receive all log INFO record notifications to our Growl client.

The solution is to use the `CallbackFilterHandler` to send only one notification (the end message only).

Suppose we want to be notified only if at least 5 files were proceeded.

Filter rules.

```
<?php

    $filters = array(
        function ($record) {
            if (!array_key_exists('count', $record['context'])) {
                return false;
            }
            return ($record['context']['count'] > 5);
        }
    );
```

And in script, we will replaced the previous `$growl` instance with this one :

```
<?php
use Bartlett\Monolog\Handler\CallbackFilterHandler;

    $growl = new CallbackFilterHandler(
        new GrowlHandler(
            array(), // with all default options
            Logger::INFO
        ),
        $filters
    );
```

Filter rules limitation is only your imagination !

6.3. Next

Read more

- `CallbackFilterHandler` is not part of standard Monolog distribution, but is available on Packagist bartlett/monolog-callbackfilterhandler [http://packagist.org/packages/bartlett/monolog-callbackfilterhandler] and as such installable via Composer. Learn more [http://php5.laurent-laville.org/callbackfilterhandler/].
- Learn how to configure your Growl Application. See Chapter 5, *Configuration*