

# Introducing CGMessageQueue

Asynchronous Programming with CMSMS

By: Robert Campbell <calguy1000@gmail.com>

Last Edited: 30. Dec. 2015

© 2015 by Robert Campbell – All Rights Reserved

## Who Should Read this Document:

This document is aimed at experienced PHP programmers who need a simple method for creating tasks that can be handled asynchronously, without the site visitor having to wait for their completion.

Readers should have significant experience with PHP, particularly with object oriented programming. And with creating a CMSMS module.

## What is a Message Queue:

In simple terms, a message queue is just that a list of messages. The messages are organized in a First-In-First-Out (FIFO). I.e: the messages are removed from the list and processed in the order in which they were added. In the computer science world message queues are a method of interprocess communication.

Messages usually contain data that must be 'handled' or 'processed' in some way. The other processes must have a way of understanding and dealing with these messages.

This module provides a simple base class to encapsulate the data of the message, the handling routine and all of the extra data that the system needs for handling the message. This allows one handler routine to handle many different types of messages.

## When to use Asynchronous Programming

This type of asynchronous programming is ideally suited for small tasks that may need to be triggered by some user, may take a bit of time to process, or may be heavy on the server, and that the site visitor does not need to wait for.

Examples of usage are:

- Creating a user account, and emailing the credentials after registration
- Importing a CSV file after it has been uploaded
- Emailing user confirmation after an online purchase
- Emailing a group of users after some data has changed in some way.

Basically, any task for which we have all of the necessary data, that the user does not necessarily need to be present for, should only take a few seconds to complete, and can happen 'in a few minutes time' without a problem.

In a website environment there are many tasks ideally suited for a message queue.

## Features of this Module:

- A single base class to encapsulate the data and the message handler

This allows a single processing routine to handle multiple different types of messages.

- Message handlers can be embedded into third party modules
- Multiple different queues are possible

For situations where different types of messages can be handled at different priorities multiple processing jobs can be created for different 'queues'. This allows finer grain control over server resources.

- Multiple messages can be processed in one call to the process action (depending upon time)

The process action of the CGMessageQueue module will continually pop messages off of the list, and process them until either the queue is empty, or a preset time interval (30 seconds by default) has expired.

- Messages can expire if not already handled

## A Simple Example

In the administration panel for this module is a simple form to allow entering an email address that will asynchronously send a fixed text email to the email address provided. This section will describe the code for that test.

First, we create a message class:

```
<?php
namespace CGMessageQueue;

class TestEmailMessage extends MessageBase
{
    private $_addr;

    public function __construct($addr)
    {
        parent::__construct();
        $this->set_module('CGMessageQueue');
        $this->set_queue('TestEmailMessage');
        $this->_addr = trim($addr);
    }

    public function handle()
    {
        $mailer = new \cms_mailer;
        $mailer->AddAddress($this->_addr);
        $mailer->SetSubject('Test from CGMessageQueue');
        $mailer->SetBody('This is a test message from CGMessageQueue');
        $mailer->Send();
        debug_to_log(__METHOD__.' message sent');
    }
}
```

As previously mentioned, this test class is the sample provided with the CGMessageQueue module. It is located within the modules/CGMessageQueue/lib directory of your CMSMS install and is using the CGMessageQueue namespace.

This class extends the MessageBase class and adds a \$\_addr property to contain the email address that we will be sending to.

In the constructor of this method we first call the base class' constructor, and then set the module and queue for processing to some hardcoded values. We then set the \$\_addr property of the object.

The handle method is where all of the processing of the task is performed. In this example case we create a new cmsmailer object, add an address, a subject, and a message body and then send the message. Lastly we send a debug message to the log file.

This is just the class definition. It determines the data associated with the message, and how the message will be processed. Next, we need to add an object of this type to the message queue.

This is the code that does that (from the action.defaultadmin.php file):

```
$test_msg = new \CGMessageQueue\TestEmailMessage($email);  
$test_msg->send();
```

Here we merely create a new object of type TestEmailMessage, and provide it with the email address we specified in the form. Lastly, to add the message to the queue for later processing, we 'send' the message.

Depending upon other items in the queue, and how often the processing is performed the recipient should receive that email within a few minutes.

This very trivial example illustrates the important concepts of the CGMessageQueue module:

- A simple way to encapsulate data and handling code for asynchronous processing
- Performing asynchronous processing.

## Integration into your CMSMS Module

To use this module for asynchronous processing tasks of your module you must:

- Configure the module processing cron job as specified within the module's help
- Add CGMessageQueue as a dependency of your module
  - Edit the GetDependencies() method of your module

```
public function GetDependencies() {  
    return array('CGMessageQueue'=>'1.0');  
}
```

- Bump your module's version number:

```
public function GetVersion() { return '0.1.1'; }
```

Every time you alter the dependencies of your module you must bump the version number, and go through the upgrade process so that the dependencies are stored properly in the CMSMS database.

- Create a class that is derived from \CGMessageQueue\MessageBase
- Create an object of that class, and 'send' it to the message queue.

## Conclusion

This document is a simple introduction to asynchronous programming within CMS Made Simple, and to the CGMessageQueue library. For further information, please feel free to submit a question in the appropriate section of the [CMSMS forum](#).

## References

- CGMessageQueue Reference Documentation: <http://apidoc.calguy1000.com/CGMessageQueue>
- CGExtensions Reference Documentation: <http://apidoc.calguy1000.com/CGExtensions>
- CMSMS API Documentation: <http://apidoc.cmsmadesimple.org>
- Talks about Message Queues: [https://en.wikipedia.org/wiki/Message\\_queue](https://en.wikipedia.org/wiki/Message_queue)