# Zend_Log_Writer_Mail

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

## Zend Framework: Zend_Log_Writer_Mail Component Proposal

| | |
|---|---|
| **Proposed Component Name** | Zend_Log_Writer_Mail |
| **Developer Notes** | http://framework.zend.com/wiki/display/ZFDEV/Zend_Log_Writer_Mail |
| **Proposers** | Brian DeShong<br>Matthew Weier O'Phinney (Zend Liaison) |
| **Revision** | 1.0 - 26 January 2008 - Initial creation (wiki revision: 25) |

## Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

## 1. Overview

Zend_Log_Writer_Mail is a Zend_Log writer for sending log entries to recipient(s) via email.

Proposal of this class is motivated by use in a batch script environment, where logs need to be kept, but developers also need to be notified via email of any notices, warnings, errors, etc.; will be illustrated below in "Use Cases."

## 2. References

- Original Zend_Log rewrite proposal
  - Note how mention is made of a Mail-based log writer, but one does not exist at this time.

## 3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** attempt to deliver appropriate log entries to given recipient(s)
- This component **will** adhere to standard Zend_Log_Writer formatting and filtering conventions
- This component **will** allow for a minimum of a plaintext message body, with an optional HTML message body
- This component **will** dynamically set mail subject to include the number of entries that occurred per-priority level if the caller so chooses
- This component **will not** open connections to the Zend_Mail transport until a message is ready to be sent

- This component **will not** allow for on-demand mailing of log entries

## 4. Dependencies on Other Framework Components

- Zend_Mail
- Zend_Log_Writer_Abstract
- Zend_Log_Exception
- Zend_Layout

## 5. Theory of Operation

User instantiates a Zend_Mail object and populates it with data for recipients that should be notified of any log entries. Zend_Mail object is then passed to constructor for Zend_Log_Writer_Mail object.

Optionally, user may instantiate an instance of Zend_Layout to provide an HTML-based message body.

Zend_Log_Writer_Mail::_write() builds up an array of log entry lines to use as the body of the email message to the recipients.

Email should not be sent upon the call of Zend_Log_Writer_Mail::_write(); the email should be sent upon the call to shutdown() if there are log entry lines to use in the body.

Once email has been sent to recipients, reference to Zend_Mail object is NOT removed as the user may want to continue using it.

## 6. Milestones / Tasks

- Milestone 1: [DONE] Initial class code is drafted for proposal
- Milestone 2: [DONE] Proposal finalized and readied for review
- Milestone 3: Working prototype checked in to incubator for community review
- Milestone 3: Unit tests are created with 90%+ code coverage (assume that we can't test the actual receiving of email?)
- Milestone 4: Initial documentation completed

## 7. Class Index

- Zend_Log_Writer_Mail

## 8. Use Cases

**UC-01**

```
$log = new Zend_Log();
$log->addWriter(new Zend_Log_Writer_Stream('php://stdout'));

$mail = new Zend_Mail();
$mail->setDefaultTransport(new Zend_Mail_Transport_Smtp());

$mail->setFrom('foo@example.org');
$mail->addTo('techlead@example.org');
$mail->addTo('devteam@example.org');
$mail->setSubject('Video transcoding -- ERRORS');

$mailWriter = new Zend_Log_Writer_Mail($mail);

// Email warnings and higher to recipients.
$mailWriter->addFilter(Zend_Log::WARN);

$log->addWriter($mailWriter);
...
$log->info('processing item X');
$log->err('failed to connect to video transcoding service');
$log->info('shutting down');

// Recipients only receive the err level message in their email.
// All log entries are written to STDOUT.
```

**UC-02**

```
$log = new Zend_Log();
$log->addWriter(new Zend_Log_Writer_Stream('php://stdout'));

$mail = new Zend_Mail();
$mail->setDefaultTransport(new Zend_Mail_Transport_Smtp());

$mail->setFrom('foo@example.org');
$mail->addTo('devteam@example.org');
$mail->setSubject('Video transcoding -- ERRORS');

$mailWriter = new Zend_Log_Writer_Mail($mail);

// Email warnings and higher to recipients.
$mailWriter->addFilter(Zend_Log::WARN);

$log->addWriter($mailWriter);

// Load up $items.

// Process items.
foreach ($items as $item) {
    $log->info('processing item X');
    // Process item
    $log->info('completed processing item X');
}

$log->info('shutting down');

// Only info messages were logged, so no email is sent to recipients.
// All log entries are written to STDOUT.
```

**UC-03**

```
$log = new Zend_Log();
$log->addWriter(new Zend_Log_Writer_Stream('php://stdout'));

$mail = new Zend_Mail();
$mail->setDefaultTransport(new Zend_Mail_Transport_Smtp());

$mail->setFrom('foo@example.org');
$mail->addTo('devteam@example.org');
$mail->setSubject('Video transcoding -- ERRORS');

$mailWriter = new Zend_Log_Writer_Mail($mail);

// Email warnings and higher to recipients.
$mailWriter->addFilter(Zend_Log::WARN);

$log->addWriter($mailWriter);

// Load up $items.

// Process items.
foreach ($items as $item) {
    try {
        $log->info('processing item X');
        // Process item
        $log->info('completed processing item X');
    } catch (Foo_Exception $e) {
        $log->error(
            'item X; ' .
                'caught exception; ' .
                'continuing to next item; ' .
                'message = ' . $e->getMessage() . '; ' .
                'code = ' . $e->getCode() . '; ' .
                'trace = ' . $e->getTraceAsString());
    }
}

$log->info('shutting down');

// If any exceptions occurred, they are sent to recipients in email body.
// All log entries are written to STDOUT.
```

**UC-04**

```
$mail = new Zend_Mail();
$mail->setDefaultTransport(new Zend_Mail_Transport_Smtp());
$mail->setFrom('foo@example.org');
$mail->addTo('techlead@example.org');
$mail->addTo('devteam@example.org');
$mail->setSubject('Video transcoding -- ERRORS');

// Use default HTML layout.
$layout = new Zend_Layout();
$layout->setLayoutPath('/path/to/layouts');
$layout->setLayout('default');

// Instantiate writer with use of layout.
$mailWriter = new Zend_Log_Writer_Mail($mail, $layout);

// Email warnings and higher to recipients.
$mailWriter->addFilter(Zend_Log::WARN);

// Create instance of Zend_Log.
$log = new Zend_Log();
$log->addWriter($mailWriter);
...
$log->info('processing item X');
$log->err('failed to connect to video transcoding service');
$log->info('shutting down');
```

**UC-05**

```
$mail = new Zend_Mail();
$mail->setDefaultTransport(new Zend_Mail_Transport_Smtp());
$mail->setFrom('foo@example.org');
$mail->addTo('techlead@example.org');
$mail->addTo('devteam@example.org');
// Note that a subject is NOT set on Zend_Mail object

// Use default HTML layout.
$layout = new Zend_Layout();
$layout->setLayoutPath('/path/to/layouts');
$layout->setLayout('default');

// Instantiate writer with use of layout.
$mailWriter = new Zend_Log_Writer_Mail($mail, $layout);

// Email warnings and higher to recipients.
$mailWriter->addFilter(Zend_Log::WARN);

// Set subject prepend text; entry counts will be appended
// to the mail subject string upon shutdown() of the writer.
$mailWriter->setSubjectPrependText('Video transcoding -- ERRORS');

// Create instance of Zend_Log.
$log = new Zend_Log();
$log->addWriter($mailWriter);
...
$log->info('processing item X');
$log->err('failed to connect to video transcoding service');
$log->info('shutting down');
```

## 9. Class Skeletons

```
<?php
```

```
/**
 * Zend Framework
 *
 * LICENSE
 *
 * This source file is subject to the new BSD license that is bundled
 * with this package in the file LICENSE.txt.
 * It is also available through the world-wide-web at this URL:
 * http://framework.zend.com/license/new-bsd
 * If you did not receive a copy of the license and are unable to
 * obtain it through the world-wide-web, please send an email
 * to license@zend.com so we can send you a copy immediately.
 *
 * @category   Zend
 * @package    Zend_Log
 * @subpackage Writer
 * @copyright  Copyright (c) 2005-2008 Zend Technologies USA Inc. (http://www.zend.com)
 * @license    http://framework.zend.com/license/new-bsd     New BSD License
 * @version    $Id: Mail.php 217 2008-11-16 03:51:54Z brian $
 */

/** Zend_Log_Writer_Abstract */
require_once 'Zend/Log/Writer/Abstract.php';

/** Zend_Log_Exception */
require_once 'Zend/Log/Exception.php';

/**
 * Class used for writing log messages to email via Zend_Mail.
 *
 * Allows for emailing log messages at and above a certain level via a
 * Zend_Mail object.  Note that this class only sends the email upon
 * completion, so any log entries accumulated are sent in a single email.
 *
 * @category Zend
 * @package Zend_Log
 * @subpackage Writer
 * @copyright Copyright (c) 2005-2008 Zend Technologies USA Inc. (http://www.zend.com)
 * @license http://framework.zend.com/license/new-bsd     New BSD License
 * @version $Id: Mail.php 217 2008-11-16 03:51:54Z brian $
 */
class Zend_Log_Writer_Mail extends Zend_Log_Writer_Abstract
{
    /**
     * Array of formatted events to include in message body.
     *
     * @var array
     */
    private $_eventsToMail = array();

    /**
     * Array of formatted lines for use in an HTML email body; these events
     * are formatted with an optional formatter if the caller is using
     * Zend_Layout.
     *
     * @var array
     */
    private $_layoutEventsToMail = array();

    /**
     * Zend_Mail instance to use
     *
     * @var Zend_Mail
     */
    private $_mail;

    /**
     * Zend_Layout instance to use; optional.
     *
     * @var Zend_Layout
```

```
     */
    private $_layout;

    /**
     * Optional formatter for use when rendering with Zend_Layout.
     *
     * @var Zend_Log_Formatter_Interface
     */
    private $_layoutFormatter;

    /**
     * Array keeping track of the number of entries per priority level.
     *
     * @var array
     */
    private $_numEntriesPerPriority = array();

    /**
     * Subject prepend text.
     *
     * Can only be used of the Zend_Mail object has not already had its
     * subject line set.  Using this will cause the subject to have the entry
     * counts per-priority level appended to it.
     *
     * @var string|null
     */
    private $_subjectPrependText;

    /**
     * Class constructor.
     *
     * Constructs the mail writer; requires a Zend_Mail instance, and takes an
     * optional Zend_Layout instance.  If Zend_Layout is being used,
     * $this->_layout->events will be set for use in the layout template.
     *
     * @param Zend_Mail $mail Mail instance
     * @param Zend_Layout $layout Layout instance; optional
     * @return void
     */
    public function __construct(Zend_Mail $mail, Zend_Layout $layout = null)
    {
        $this->_mail = $mail;
        $this->_layout = $layout;
        $this->_formatter = new Zend_Log_Formatter_Simple();
    }

    /**
     * Places event line into array of lines to be used as message body.
     *
     * Handles the formatting of both plaintext entries, as well as those
     * rendered with Zend_Layout.
     *
     * @param array $event Event data
     * @return void
     */
    protected function _write($event)
    {
        // Track the number of entries per priority level.
        if (!isset($this->_numEntriesPerPriority[$event['priorityName']])) {
            $this->_numEntriesPerPriority[$event['priorityName']] = 1;
        } else {
            $this->_numEntriesPerPriority[$event['priorityName']]++;
        }

        $formattedEvent = $this->_formatter->format($event);

        // All plaintext events are to use the standard formatter.
        $this->_eventsToMail[] = $formattedEvent;

        // If we have a Zend_Layout instance, use a specific formatter for the
```

```
        // layout if one exists.  Otherwise, just use the event with its
        // default format.
        if ($this->_layout) {
            if ($this->_layoutFormatter) {
                $this->_layoutEventsToMail[] =
                    $this->_layoutFormatter->format($event);
            } else {
                $this->_layoutEventsToMail[] = $formattedEvent;
            }
        }
    }

    /**
     * Sets a specific formatter for use with Zend_Layout events.
     *
     * Allows use of a second formatter on lines that will be rendered with
     * Zend_Layout.  In the event that Zend_Layout is not being used, this
     * formatter cannot be set, so an exception will be thrown.
     *
     * @param Zend_Log_Formatter_Interface $formatter
     * @return void
     * @throw Zend_Log_Exception
     */
    public function setLayoutFormatter(Zend_Log_Formatter_Interface $formatter)
    {
        if (!$this->_layout) {
            throw new Zend_Log_Exception(
                'cannot set formatter for layout; ' .
                    'a Zend_Layout instance is not in use');
        }

        $this->_layoutFormatter = $formatter;
    }

    /**
     * Allows caller to have the mail subject dynamically set to contain the
     * entry counts per-priority level.
     *
     * Sets the text for use in the subject, with entry counts per-priority
     * level appended to the end.  Since a Zend_Mail subject can only be set
     * once, this method cannot be used if the Zend_Mail object already has a
     * subject set.
     *
     * @param string $subject Subject prepend text.
     * @return void
     */
    public function setSubjectPrependText($subject)
    {
        if ($this->_mail->getSubject()) {
            throw new Zend_Log_Exception(
                'subject already set on mail; ' .
                    'cannot set subject prepend text');
        }

        $this->_subjectPrependText = (string) $subject;
    }

    /**
     * Sends mail to recipient(s) if log entries are present.  Note that both
     * plaintext and HTML portions of email are handled here.
     *
     * @return void
     */
    public function shutdown()
    {
        // If there are events to mail, use them as message body.  Otherwise,
        // there is no mail to be sent.
        if (empty($this->_eventsToMail)) {
            return;
        }
```

```
        if ($this->_subjectPrependText !== null) {
            // Tack on the summary of entries per-priority to the subject
            // line and set it on the Zend_Mail object.
            $numEntries = $this->_getFormattedNumEntriesPerPriority();
            $this->_mail->setSubject(
                "{$this->_subjectPrependText} ({$numEntries})");
        }


        // Always provide events to mail as plaintext.
        $this->_mail->setBodyText(implode('', $this->_eventsToMail));

        // If a Zend_Layout instance is being used, set its "messages"
        // value to the lines formatted for use with the layout.
        if ($this->_layout) {
            // Set the required "events" value for the layout.  Here we
            // are assuming that the layout is for use with HTML.
            $this->_layout->events =
                implode('', $this->_layoutEventsToMail);
            $this->_mail->setBodyHtml($this->_layout->render());
        }

        // Finally, send the mail, but re-throw any exceptions at the
        // proper level of abstraction.
        try {
            $this->_mail->send();
        } catch (Exception $e) {
            throw new Zend_Log_Exception(
                $e->getMessage(),
                $e->getCode());
        }
    }

    /**
     * Gets a string of number of entries per-priority level that occurred, or
     * null if none occurred.
     *
     * @return string|null
     */
    private function _getFormattedNumEntriesPerPriority()
    {
        if (empty($this->_numEntriesPerPriority)) {
            return null;
        }

        $strings = array();

        foreach ($this->_numEntriesPerPriority as $priority => $numEntries) {
            $strings[] = "{$priority}={$numEntries}";
        }

        return implode(', ', $strings);
    }
}
```

]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>