phpWatch 2 User Guide

Aaron M. Rosenfeld

April 16, 2010

1 Introduction

This document briefly describes the functionality of phpWatch 2 and how end users can fully utilize the system. It is not a necessity to read this document as phpWatch is fairly simple to use, but it is recommended for non-technical users or those who haven't used past version of phpWatch.

2 Support

phpWatch is driven mainly by user suggestions and requests. If there is something you would like to see in a future release or find a bug, please let the developers know! The best method of contact is through Sourceforge:

`http://sourceforge.net/projects/phpwatch/support`

If you'd like to use e-mail, please send a message to:

`developers [at] phpwatch.net`

Finally, if you find phpWatch useful please consider donating to the starving college students that develop it!

`http://phpwatch.net/donate`

3 System Purpose

phpWatch is designed to be a complete service monitoring system. It has capabilities of querying services in unique ways both through the build in features and through extension. Furthermore, phpWatch provides facilities for notifying the proper individuals by various means when a service is determined to be offline or malfunctioning. Additionally, phpWatch collects basic statistics on uptime for each service being monitored. This aspect of the system will be extended for more robust statistics in the future.

4 System Components

phpWatch is comprised of three major components: *monitors*, *contacts*, and *channels*. *Monitors* handle the querying of services, determine if responses from services indicate a problem, and dispatch notifications when there is a problem. For example, a monitor may simply attempt to open a connection to a service where another may validate a response from the service.

When a notifications is dispatched, it is sent to one or more *contacts* via one or more *channels*. A *channel* is a method by which notifications are delivered. Examples include e-mail and text-message. A *contact* is an aggregation of *channels* related to a certain individual. For example, a contact may have an associated e-mail channel and text-message channel. Each channel can be used to send notifications for any number of monitors. That is, multiple monitors can alert individuals via the same channel.

5 Automatic Querying

Monitors query their associated services only be two invocation methods. First, the "Re-query" links on the monitor page on the frontend. These are mainly for urgent matters, though. The primary method that should be setup is a cronjob (or scheduled task in Windows). As described in `README`, setting this up is relatively simple on a Linux based machine. Execute `crontab -e` as root. Add to the bottom of the file the line:

```
*/N * * * * php /path/to/phpwatch/root/directory/cron.php
```

where N is a number indicating the number of minutes between each execution. For example, to have services checked every 5 minutes, add the line:

```
*/5 * * * * php /path/to/phpwatch/root/directory/cron.php
```

To have the cronjob run at an interval other than every N minutes, refer to the Wikipedia entry at: *http://en.wikipedia.org/wiki/Cron#Overview.*

6 The Frontend

The phpWatch frontend is designed to be optional, although the vast majority of users choose to use it. It can be accessed by visiting the phpWatch directory in a web browser. On it there is one page listing all of the monitors in the system and another for managing contacts. Both can be added and modified simply.

6.1 Creating a Contact / Channel Maintenance

Contacts can be added in the contacts page. Select "New Contact" at the top and a new page will be displayed presenting a "Name" field. Fill this value in and press "Submit". To add contact information (channels) to the contact, after submitting the form, select the link that returns to the contact just added. Channels can then be added under the "Name" field.

Select a channel type and press "Add Notification Channel". The page that is displayed will ask for information specific to that channel type (e.g. e-mail address for Email channel). Pressing "Submit" after filling this information in will add the channel to the contact. Editing a contact will also allow new channels to be added and existing ones to be edited or deleted.

6.2 Creating a Monitor

To create a monitor, navigate to the monitors page, select a monitor type from the drop-down at the top left, and click "New Monitor". There will be a number of fields under "Generic Settings" which are common to all monitor types:

- **Hostname:** The hostname or IP address of service to be monitored. Keep in mind this will be resolved from the machine on which phpWatch runs so NATs and other network setups should be considered.
- **Port:** The port on which the service to be monitored runs.
- **Alias:** An optional name for the monitor. It is strongly recommended that this is *not* left blank, though, to assure notification messages are easily readable.
- **Fail Threshold:** An integer value greater than or equal to unity. The service being monitored must fail this many times *sequentially* for notifications to be sent. For example, entering "2" will cause the monitor to only send notifications after the monitor is found to be offline two *sequential* queries.

Additionally, all monitors have a "state". This allows for scheduled downtime and for monitors to be indefinitely paused. The options for state are:

- **Running:** The default state wherein uptime information is logged, the service is queried as determined by the cronjob, and notifications shall be sent as necessary.
- **Paused:** The monitor shall not query, collect statistics, nor send notifications indefinitely.
- **Scheduled Downtime:** The same as paused except a start time and duration must be specified. At the start time, the monitor will be placed in the paused state for the specified duration[1].

Beyond these settings, each type of monitor may require additional parameters to be specified. These will appear below the generic fields. Finally, at the bottom of the page, contact channels may be selected indicating which contacts should be notified in case of a service outage and by what means (channel).

[1]More accurately, it will be in the paused state for no *less* time than the specified duration and will be returned to the running state on the next cronjob execution