# WordPress Automated Maintenance System

Version: 1.0

Last Updated: December 2024

# Overview

This system provides automated maintenance for WordPress sites including plugin updates, theme updates, cache management, error log monitoring, and performance metrics tracking. The system sends detailed HTML email reports after each maintenance run.

Table of Contents

1. [Features](#features)

2. [Installation](#installation)

3. [Configuration](#configuration)

4. [Usage](#usage)

5. [Customization](#customization)

6. [Troubleshooting](#troubleshooting)

## Features

- Automated plugin updates

- Automated theme updates

- Cache clearing for popular caching plugins

- Error log monitoring

- Performance metrics tracking

- Detailed HTML email reports

- Manual run option via admin panel

- Dashboard widget for status monitoring

## Installation

### Step 1: Create Plugin Structure

1. Create a new directory in `wp-content/plugins` called `wp-maintenance-scheduler`

2. Inside this directory, create these files:

- `wp-maintenance-scheduler.php` (main plugin file)

- `class-maintenance.php` (maintenance class)

- `admin-page.php` (admin interface)

Folder Structure:

wp-content/plugins/wp-maintenance-scheduler/

├── wp-maintenance-scheduler.php

├── admin/

│ └── admin-page.php

└── includes/

└── class-maintenance.php

### Step 2: Install Required Files

1. Copy the provided code into each respective file

2. Activate the plugin through WordPress admin panel

3. Configure email settings

4. Test the system

## Configuration

### Email Settings

1. Default recipient is the WordPress admin email

2. To change recipient, use the filter:

```php

add\_filter('maintenance\_report\_email', function($email) {

return 'your-email@domain.com';

});

```

### Schedule Settings

1. Default schedule is weekly

2. To change frequency, modify the cron schedule:

```php

add\_filter('maintenance\_schedule', function($schedule) {

return 'daily'; // Options: hourly, daily, weekly

});

```

## Usage

### Automatic Maintenance

The system runs automatically based on the configured schedule. No manual intervention required.

### Manual Maintenance

1. Go to Tools → Manual Maintenance in WordPress admin

2. Click "Run Maintenance Now" button

3. Wait for completion notification

4. Check email for report

### Dashboard Widget

Displays:

- Last maintenance run time

- Next scheduled run

- Latest performance metrics

## Customization

### Adding Custom Checks

1. Extend the MaintenanceTask class

2. Add new methods for custom checks

3. Register new tasks in the run\_maintenance method

### Modifying Email Template

1. Locate the generate\_html\_report method

2. Modify HTML structure and styling

3. Add or remove sections as needed

## Troubleshooting

### Common Issues

1. Emails not sending:

- Check WordPress mail configuration

- Verify admin email address

- Check spam folder

2. Updates failing:

- Check file permissions

- Verify WordPress version compatibility

- Check server resources

3. Schedule not running:

- Verify WP-Cron is enabled

- Check server cron configuration

- Monitor error logs

## Code Implementation

Create the following files with their respective contents:

1. Main Plugin File (`wp-maintenance-scheduler.php`):

```php

<?php

/\*

Plugin Name: WP Maintenance Scheduler

Description: Automated maintenance system with email reporting

Version: 1.0

Author: Your Name

\*/

// Prevent direct access

if (!defined('ABSPATH')) {

exit;

}

// Include required files

require\_once plugin\_dir\_path(\_\_FILE\_\_) . 'class-maintenance.php';

// Initialize the plugin

function initialize\_maintenance\_scheduler() {

// Schedule maintenance tasks

if (!wp\_next\_scheduled('maintenance\_scheduler\_hook')) {

wp\_schedule\_event(time(), 'weekly', 'maintenance\_scheduler\_hook');

}

// Add admin page

require\_once plugin\_dir\_path(\_\_FILE\_\_) . 'admin-page.php';

}

register\_activation\_hook(\_\_FILE\_\_, 'initialize\_maintenance\_scheduler');

// Clean up on deactivation

function cleanup\_maintenance\_scheduler() {

wp\_clear\_scheduled\_hook('maintenance\_scheduler\_hook');

}

register\_deactivation\_hook(\_\_FILE\_\_, 'cleanup\_maintenance\_scheduler');

```

2. Class File (`class-maintenance.php`):

[Previous maintenance class code goes here with all methods]

3. Admin Page (`admin-page.php`):

```php

<?php

// Add admin menu item

add\_action('admin\_menu', 'maintenance\_scheduler\_menu');

function maintenance\_scheduler\_menu() {

add\_management\_page(

'Maintenance Scheduler',

'Maintenance Scheduler',

'manage\_options',

'maintenance-scheduler',

'maintenance\_scheduler\_page'

);

}

function maintenance\_scheduler\_page() {

?>

<div class="wrap">

<h1>WordPress Maintenance Scheduler</h1>

<div class="card">

<h2>Status</h2>

<p>Last Run: <?php echo get\_option('last\_maintenance\_run') ? date('Y-m-d H:i:s', get\_option('last\_maintenance\_run')) : 'Never'; ?></p>

<p>Next Run: <?php echo wp\_next\_scheduled('maintenance\_scheduler\_hook') ? date('Y-m-d H:i:s', wp\_next\_scheduled('maintenance\_scheduler\_hook')) : 'Not scheduled'; ?></p>

</div>

<div class="card">

<h2>Manual Run</h2>

<form method="post">

<?php wp\_nonce\_field('run\_maintenance\_nonce'); ?>

<input type="submit" name="run\_maintenance" class="button button-primary" value="Run Maintenance Now">

</form>

</div>

</div>

<?php

// Handle manual run

if (isset($\_POST['run\_maintenance']) && check\_admin\_referer('run\_maintenance\_nonce')) {

$maintenance = new SiteMaintenance();

$maintenance->run\_maintenance();

echo '<div class="notice notice-success"><p>Maintenance completed. Check your email for the report.</p></div>';

}

}

```

## Testing Steps

1. Initial Setup Test

```php

// Add to functions.php temporarily

add\_action('init', function() {

if(isset($\_GET['test\_maintenance']) && current\_user\_can('manage\_options')) {

$maintenance = new SiteMaintenance();

$maintenance->run\_maintenance();

die('Test completed - check your email');

}

});

```

2. Verify Installation:

- Check plugin appears in WordPress admin

- Verify scheduled task is created

- Confirm email settings

3. Test Manual Run:

- Use admin interface to trigger maintenance

- Verify email report is received

- Check all sections of report

## Maintenance

### Regular Tasks

1. Monitor error logs

2. Check email deliverability

3. Verify schedule is running

4. Update plugin as needed

### Updates

1. Backup before updating

2. Test in staging environment

3. Monitor first run after updates

4. Document any changes

## Security Considerations

1. File Permissions:

- Plugin files: 644

- Plugin directories: 755

2. Access Control:

- Only administrators can access

- Nonce verification on forms

- Capability checking

3. Data Handling:

- Sanitize all output

- Validate all inputs

- Use WordPress security functions

## Support

For issues or questions:

1. Check documentation first

2. Review error logs

3. Test in staging environment

4. Contact support with details

---

Remember to:

- Always backup before making changes

- Test in staging environment first

- Monitor first few runs closely

- Keep plugin updated