Using the MainWP API Client PHP

- Quick Start for Experienced Developers
- Installation
- Configuration
- Working with Sites
  - Get All Sites
  - Get a Specific Site
  - Create a Site
  - Update a Site
  - Delete a Site
- Executing Actions on Sites
  - Core Updates
  - Plugin Management
  - Theme Management
  - Backup Management
  - Post and Page Management
- Error Handling
- Advanced Usage
  - Custom Requests
  - Batch Operations
  - Pagination Handling
- Integration with MainWP Extensions
- Real-World Example: Site Health Monitor
- Conclusion
- Related Resources

This guide covers how to use the official MainWP API Client PHP library to interact with the MainWP Dashboard API. This client library simplifies the process of connecting to and working with MainWP-managed sites programmatically.

Quick Start for Experienced Developers

```
// Install via Composer
// composer require mainwp/mainwp-api-client-php

// Initialize the client
require_once 'vendor/autoload.php';

use MainWP\ApiClient\MainWP;

// Create a new client instance
$mainwp = new MainWP([
    'baseUrl' => 'https://your-mainwp-dashboard.com',
    'apiKey' => 'your-api-key',
    'apiSecret' => 'your-api-secret'
]);

// Get all managed sites
$sites = $mainwp->getSites();

// Get a specific site by ID
$site = $mainwp->getSite(123);

// Update a site
$mainwp->updateSite(123, [
    'name' => 'Updated Site Name'
]);

// Execute an action on a site
$result = $mainwp->executeAction(123, 'core_update');
```

```
// Error handling
try {
    $sites = $mainwp->getSites();
} catch (\MainWP\ApiClient\Exception\ApiException $e) {
    echo 'API Error: ' . $e->getMessage();
} catch (\MainWP\ApiClient\Exception\AuthenticationException $e) {
    echo 'Authentication Error: ' . $e->getMessage();
} catch (\Exception $e) {
    echo 'Error: ' . $e->getMessage();
}
```

Installation

The MainWP API Client PHP library can be installed via Composer:

```
composer require mainwp/mainwp-api-client-php
```

If you're not using Composer, you can download the library from GitHub and include it manually.

Configuration

Before using the API client, you need to set up API access in your MainWP Dashboard:

1. Go to your MainWP Dashboard
2. Navigate to Settings > API
3. Enable the API
4. Create a new API key and secret
5. Note the API key and secret for use with the client

Then, initialize the client with your MainWP Dashboard URL, API key, and API secret:

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

use MainWP\ApiClient\MainWP;

$mainwp = new MainWP([
    'baseUrl' => 'https://your-mainwp-dashboard.com',
```

```
    'apiKey' => 'your-api-key',
    'apiSecret' => 'your-api-secret'
]);
```

Working with Sites

The MainWP API Client provides several methods for working with managed sites:

Get All Sites

```
// Get all sites
$sites = $mainwp->getSites();

// Get sites with specific tags
$sites = $mainwp->getSites(['tags' => ['production', 'client-a']]);

// Get sites with pagination
$sites = $mainwp->getSites(['page' => 1, 'per_page' => 10]);
```

Get a Specific Site

```
// Get site by ID
$site = $mainwp->getSite(123);

// Get site by URL
$site = $mainwp->getSiteByUrl('https://example.com');
```

Create a Site

```
$site = $mainwp->createSite([
    'url' => 'https://example.com',
    'name' => 'Example Site',
    'adminname' => 'admin',
    'adminpass' => 'password',
    'tags' => ['client-a', 'production']
```

```
]);
```

Update a Site

```
$mainwp->updateSite(123, [
    'name' => 'Updated Site Name',
    'tags' => ['client-a', 'staging']
]);
```

Delete a Site

```
$mainwp->deleteSite(123);
```

Executing Actions on Sites

You can execute various actions on managed sites:

Core Updates

```
// Update WordPress core
$result = $mainwp->executeAction(123, 'core_update');
```

Plugin Management

```
// Install a plugin
$result = $mainwp->executeAction(123, 'plugin_install', [
    'plugin' => 'akismet'
]);

// Activate a plugin
$result = $mainwp->executeAction(123, 'plugin_activate', [
    'plugin' => 'akismet/akismet.php'
]);
```

```
// Update a plugin
$result = $mainwp->executeAction(123, 'plugin_update', [
    'plugin' => 'akismet/akismet.php'
]);

// Deactivate a plugin
$result = $mainwp->executeAction(123, 'plugin_deactivate', [
    'plugin' => 'akismet/akismet.php'
]);

// Delete a plugin
$result = $mainwp->executeAction(123, 'plugin_delete', [
    'plugin' => 'akismet/akismet.php'
]);
```

Theme Management

```
// Install a theme
$result = $mainwp->executeAction(123, 'theme_install', [
    'theme' => 'twentytwentyone'
]);

// Activate a theme
$result = $mainwp->executeAction(123, 'theme_activate', [
    'theme' => 'twentytwentyone'
]);

// Update a theme
$result = $mainwp->executeAction(123, 'theme_update', [
    'theme' => 'twentytwentyone'
]);

// Delete a theme
$result = $mainwp->executeAction(123, 'theme_delete', [
    'theme' => 'twentytwentyone'
]);
```

Backup Management

```
// Create a backup
$result = $mainwp->executeAction(123, 'backup_create', [
    'type' => 'full', // 'full', 'db', or 'files'
    'destination' => 'remote', // 'remote' or 'local'
    'filename' => 'backup-' . date('Y-m-d')
]);

// Restore a backup
$result = $mainwp->executeAction(123, 'backup_restore', [
    'backup_file' => 'backup-2023-01-01.zip'
]);
```

Post and Page Management

```
// Create a post
$result = $mainwp->executeAction(123, 'post_create', [
    'title' => 'New Post',
    'content' => 'This is the content of the post.',
    'status' => 'publish',
    'categories' => ['Category 1', 'Category 2'],
    'tags' => ['tag1', 'tag2']
]);

// Update a post
$result = $mainwp->executeAction(123, 'post_update', [
    'post_id' => 456,
    'title' => 'Updated Post',
    'content' => 'This is the updated content.'
]);

// Delete a post
$result = $mainwp->executeAction(123, 'post_delete', [
    'post_id' => 456
]);
```

Error Handling

The MainWP API Client throws different types of exceptions for different error scenarios:

```
try {
    $sites = $mainwp->getSites();
} catch (\MainWP\ApiClient\Exception\ApiException $e) {
    // API-related errors (invalid parameters, resource not found, etc.)
    echo 'API Error: ' . $e->getMessage();
    echo 'Error Code: ' . $e->getCode();
} catch (\MainWP\ApiClient\Exception\AuthenticationException $e) {
    // Authentication errors (invalid API key/secret, expired token, etc.)
    echo 'Authentication Error: ' . $e->getMessage();
} catch (\MainWP\ApiClient\Exception\ConnectionException $e) {
    // Connection errors (network issues, timeout, etc.)
    echo 'Connection Error: ' . $e->getMessage();
} catch (\Exception $e) {
    // Other errors
    echo 'Error: ' . $e->getMessage();
}
```

Advanced Usage

Custom Requests

If you need to make a custom API request that isn't covered by the built-in methods:

```
// Make a GET request
$response = $mainwp->get('custom-endpoint', ['param' => 'value']);

// Make a POST request
$response = $mainwp->post('custom-endpoint', ['data' => 'value']);

// Make a PUT request
$response = $mainwp->put('custom-endpoint', ['data' => 'value']);

// Make a DELETE request
$response = $mainwp->delete('custom-endpoint');
```

Batch Operations

For performing operations on multiple sites at once:

```
// Execute the same action on multiple sites
$results = $mainwp->batchExecuteAction([123, 456, 789], 'core_update');

// Different actions on different sites
$batch = [
    ['site_id' => 123, 'action' => 'core_update'],
    ['site_id' => 456, 'action' => 'plugin_update', 'params' => ['plugin' =>
'akismet/akismet.php']],
    ['site_id' => 789, 'action' => 'theme_update', 'params' => ['theme' => 'twentytwentyone']]
];
$results = $mainwp->batchExecute($batch);
```

Pagination Handling

For working with paginated results:

```
// Get all sites with pagination
$page = 1;
$per_page = 10;
$all_sites = [];

do {
    $sites = $mainwp->getSites(['page' => $page, 'per_page' => $per_page]);
    $all_sites = array_merge($all_sites, $sites);
    $page++;
} while (count($sites) === $per_page);
```

Integration with MainWP Extensions

If you're building a MainWP extension, you can use the API client to interact with the MainWP Dashboard:

```
class My_MainWP_Extension {
    private $mainwp;

    public function __construct() {
        // Initialize the API client
        $this->mainwp = new \MainWP\ApiClient\MainWP([
            'baseUrl' => get_option('mainwp_url'),
            'apiKey' => get_option('mainwp_api_key'),
            'apiSecret' => get_option('mainwp_api_secret')
        ]);

        // Add menu items
        add_action('admin_menu', [$this, 'admin_menu']);
    }

    public function admin_menu() {
        add_menu_page(
            'My Extension',
            'My Extension',
            'manage_options',
            'my-extension',
            [$this, 'render_main_page']
        );
    }

    public function render_main_page() {
        try {
            // Get all sites
            $sites = $this->mainwp->getSites();

            // Display sites
            echo '<h1>Managed Sites</h1>';
            echo '<ul>';
            foreach ($sites as $site) {
                echo '<li>' . esc_html($site['name']) . ' (' . esc_html($site['url']) .
')</li>';
            }
            echo '</ul>';
        } catch (\Exception $e) {
```

```
            echo '<div class="error"><p>Error: ' . esc_html($e->getMessage()) . '</p></div>';
        }
    }
}

// Initialize the extension
new My_MainWP_Extension();
```

Real-World Example: Site Health Monitor

Here's a complete example of a script that checks the site health status of all managed sites and sends an email if any issues are found:

```
<?php
// site-health-monitor.php

require_once 'vendor/autoload.php';

use MainWP\ApiClient\MainWP;

// Configuration
$config = [
    'baseUrl' => 'https://your-mainwp-dashboard.com',
    'apiKey' => 'your-api-key',
    'apiSecret' => 'your-api-secret',
    'emailTo' => 'admin@example.com',
    'emailFrom' => 'monitor@example.com'
];

// Initialize the MainWP API client
$mainwp = new MainWP([
    'baseUrl' => $config['baseUrl'],
    'apiKey' => $config['apiKey'],
    'apiSecret' => $config['apiSecret']
]);
```

```
try {
    // Get all sites
    $sites = $mainwp->getSites();

    $issues = [];

    // Check each site
    foreach ($sites as $site) {
        try {
            // Get site health data
            $health = $mainwp->executeAction($site['id'], 'site_health');

            // Check for issues
            if ($health['status'] === 'critical') {
                $issues[] = [
                    'site' => $site['name'],
                    'url' => $site['url'],
                    'issues' => $health['issues']
                ];
            }
        } catch (\Exception $e) {
            // Add connection issues to the list
            $issues[] = [
                'site' => $site['name'],
                'url' => $site['url'],
                'issues' => ['Connection error: ' . $e->getMessage()]
            ];
        }
    }

    // Send email if issues were found
    if (!empty($issues)) {
        $subject = 'Site Health Issues Detected';

        $message = "The following sites have health issues:\n\n";

        foreach ($issues as $issue) {
            $message .= "Site: {$issue['site']} ({$issue['url']})\n";
```

```
            $message .= "Issues:\n";

            foreach ($issue['issues'] as $detail) {
                $message .= "- {$detail}\n";
            }

            $message .= "\n";
        }

        $headers = "From: {$config['emailFrom']}\r\n";

        mail($config['emailTo'], $subject, $message, $headers);

        echo "Email sent with " . count($issues) . " site issues.\n";
    } else {
        echo "No issues found.\n";
    }
} catch (\Exception $e) {
    echo "Error: " . $e->getMessage() . "\n";
}
```

Conclusion

The MainWP API Client PHP library provides a simple and powerful way to interact with your MainWP Dashboard programmatically. Whether you're building a MainWP extension, integrating MainWP with other systems, or automating your WordPress management workflows, this client library makes it easy to work with the MainWP API.

For more information and the latest updates, visit the MainWP API Client PHP GitHub repository.

Related Resources

- MainWP Dashboard API Documentation
- MainWP Developer Documentation
- Creating a Basic Extension
- API Integration Best Practices
- API Integration Patterns

□□□ #2
□ □□□ □□□ 2026-08-29 22:03:44 UTC
□ □□□ □□□ 2026-08-29 22:03:44 UTC