Composer Support for Private WordPress Packages

PaddlePress Pro · https://handyplugins.co/docs/paddlepress-composer-support/

PaddlePress Pro allows you to distribute your WordPress plugins and themes via Composer, using license-protected endpoints.

This is useful for developers who want to provide private, versioned Composer access to their products while enforcing licensing and update control.

Enabling Composer Support

Before you or your customers can install your product via Composer, you need to activate Composer support in your PaddlePress settings.

1. **Go to**:
   WP Admin → PaddlePress → Settings → Software Licensing
2. **Enable License Server**
   Make sure “Enable Software Licensing” is active.
3. **Enable Composer Support**
   Toggle on the “Enable Composer Support” option.
4. **Choose Download Type**
   Set your product’s download type to either:
   - **WordPress Plugin**
   - **WordPress Theme**

> Currently, only themes and plugins are supported for our Composer integration.

Composer Setup (Consumer Side)

Here's an example `composer.json` file that a customer (or yourself) can use to install the product:

```
{
  "name": "vendor/project",
  "description": "Example project",
  "type": "project",
  "minimum-stability": "dev",
  "prefer-stable": true,
  "config": {
    "platform": {
      "php": "7.4"
    },
    "allow-plugins": {
      "composer/installers": true
    }
  },
  "repositories": [
    {
      "type": "composer",
      "url": "https://yourdomain.com/wp-json/paddlepress-
api/v1/composer/{download_tag}/packages.json",
      "options": {
        "http": {
          "header": [
            "license-key: YOUR_LICENSE_KEY_HERE",
            "license-url: https://example.com"
          ]
        }
      }
    }
  ],
  "require": {
    "yourvendor/{download_tag}": "^2.5"
  },
  "extra": {
    "installer-paths": {
      "wp-content/plugins/{$name}/": [
        "type:wordpress-plugin"
```

```
      ],
      "wp-content/themes/{$name}/": [
        "type:wordpress-theme"
      ]
    }
  }
}
```

> **Note:** Composer does not currently support storing custom headers in `auth.json`. Avoid committing credentials and consider using environment variable injection or local override files for safer usage.

Supported Package Types

- WordPress Plugins (`type: wordpress-plugin`)
- WordPress Themes (`type: wordpress-theme`)

> Other Composer package types are **not currently supported**.

Example Installation

Once everything is set up:

```
composer require yourvendor/plugin-slug
```

Viewing Available Packages

Each package (download tag) has its own metadata endpoint. Composer endpoints are scoped per download tag.

```
https://example.com/wp-json/paddlepress-api/v1/composer/{download_tag}/packages.json
```

- No license key is required to browse metadata — validation only happens during the actual download.
- This endpoint is **public**, but license validation is only enforced during the **actual package download**.

Pull Latest Version

If you want to always install the latest available version of your plugin or theme, you can use the `dev-latest` version tag in your `composer.json`:

```
"require": {
  "handypluginsco/magic-login-pro": "dev-latest"
}
```

This special version alias will always resolve to the most recent version available on the update server, regardless of its semantic version number.

Changing Composer Install Folder for a Plugin

If your PaddlePress Composer **download tag** doesn't match the plugin's actual directory name, you can control where it installs by overriding the `installer-name`.

For example, if your download tag is `paddlepress`, but your plugin folder is `paddlepress-pro`, use this:

```
add_filter( 'paddlepress_composer_packages', function( $packages, $package_slug ) {
	if ( $package_slug === 'paddlepress' ) {
		foreach ( $packages as &$versions ) {
			foreach ( $versions as &$version ) {
				$version['extra']['installer-name'] = 'paddlepress-pro';
			}
		}
	}

	return $packages;
}, 10, 2 );
```

This ensures that when a user installs the plugin via Composer, it ends up in:

```
wp-content/plugins/paddlepress-pro/
```

Note: I'm using "paddlepress" for "paddlepress-pro" since previous versions are tied to the "paddlepress" download tag for the auto-updater. It's not easy to change that without breaking existing installations. The snippet above fixes this issue and allows continued use of the "paddlepress" tag without any problems.

⚠ Important Notes

- `dev-latest` is a virtual version and may not follow semantic versioning rules, so it's best used in development or CI pipelines where automatic updates are acceptable.
- Composer might cache results aggressively — run `composer clear-cache` before updates if changes don’t show up immediately.

Troubleshooting

Having trouble using Composer with your PaddlePress-powered endpoint? Here's how to diagnose and resolve common issues:

Errors

- Make sure the **license key is valid and active**.
- Confirm that **Composer support is enabled** in PaddlePress settings.
- Ensure the request includes the required headers (`license-key`, `license-url`).

Package Not Found

- By default, PaddlePress uses your domain name to generate the vendor name.
  For example:
  - `handyplugins.co` → `handypluginsco`
  - `staging.handyplugins.co` → `handypluginsco`
  - `example.co.uk` → `exampleco`
- If needed, override the vendor name using the `paddlepress_composer_vendor_name` filter.

- The final value must match **exactly** in both:
  - The package name in your `composer.json` and your download_tag
- If Composer reports "precondition failed" (412) or similar, it means your license key or license URL was invalid or missing.

Caching Issues

- Run `composer clear-cache` to clear stale package metadata or version info.

Rate Limiting

- If you've enabled rate limiting in PaddlePress, it applies to Composer endpoints as well.
- Ensure your users stay within the configured request thresholds.

Managing Large Product Lists

If you manage a large number of WordPress products, you can increase this limit using the `paddlepress_composer_packages_args` filter. By default, PaddlePress loads up to **50 downloadable items** per Composer request. (This is sufficient for most of the use cases). But you can increase using:

```
add_filter( 'paddlepress_composer_packages_args', function( $args ) {
  $args['posts_per_page'] = 100;
  return $args;
});
```

Example: Installing or Updating PaddlePress Pro via Composer

1. **Add a repository block to your composer.json**

```
"repositories": [
  {
    "type": "composer",
    "url": "https://handyplugins.co/wp-json/paddlepress-
api/v1/composer/paddlepress/packages.json",
    "options": {
      "http": {
```

```
        "header": [
          "license-key: YOUR_LICENSE_KEY_HERE",
          "license-url: https://yourdomain.com"
        ]
      }
    }
  }
],
```

2. Require the package

To install the latest available version at any time, use the dev-latest

```
composer require handypluginsco/paddlepress:dev-latest
```

This ensures that Composer always pulls the most recent published release, even if you don’t increment semantic versions traditionally (e.g. 2025.06.12.14.23).

3. Updating the package later

If it's already installed and you want to upgrade:

```
composer update handypluginsco/paddlepress
```

Composer will:

- Hit your PaddlePress-powered `packages.json` endpoint.
- Authenticate with license headers.
- Fetch and install the latest matching version.

Summary

- Use `dev-latest` to always get the newest version.
- Make sure each package has its **own** Composer repository block with valid credentials.
- License verification happens only during the actual download step.