

Block Specific Plugin Guidelines



- [Guide to submitting plugins to the Block Directory](#)
 - [User Expectations](#)
 - [Developer Expectations](#)
 - [Definitions](#)
 - [Block Plugins and the Block Directory](#)
 - [Guidelines](#)
 - [1. Block Plugins are for the Block Editor.](#)
 - [2. Block Plugins are separate blocks.](#)
 - [3. Plugin and block names should reflect the block's purpose.](#)
 - [3a. Block names should be unique and properly namespaced.](#)
 - [4. Block Plugins must include a block.json file.](#)
 - [5. Block Plugins must work independently.](#)
 - [6. Block Plugins should work seamlessly.](#)
 - [7. Server-side code should be kept to a minimum.](#)
 - [8. Must not include advertisements or promotional notices.](#)
-

[info]All Block Specific plugins must also comply with the [overall plugin guidelines](#). These additional guidelines are unique to block specific plugins.[/info]

Guide to submitting plugins to the Block Directory

The goal of the [Block Directory](#) is to provide a safe place for WordPress users to find and install new blocks.

User Expectations

- Users will have a place they can go, both within their WordPress admin and on WordPress.org, where they can download and install blocks that have been pre-vetted for major security issues.
- Users will be able to one-click install blocks from their admin, one at a time.
- Blocks will appear in their block library after installation and activation.
- Blocks will work seamlessly and immediately without intrusive advertisements or upselling.

Developer Expectations

- Developers will have a concrete set of requirements and guidelines to follow when writing blocks for the Block Directory.
- Following these guidelines will help ensure that the blocks they develop can be seamlessly installed in the editor.
- Blocks submitted to the Block Directory are held to stricter technical and policy rules than WordPress plugins in general.
- Plugins with blocks that do not meet the Block Directory Guidelines may still be submitted to the Plugin Directory.

Definitions

For the purposes of the Block Directory, we distinguish between two types of plugins:

1. Plugins that exist solely to distribute a block.
2. All other plugins, including those that bundle many independent blocks, plugins that contain blocks in addition to other functionality, and plugins with no blocks at all.

These guidelines apply specifically to the first type of plugin, which is called a Block Plugin. If a plugin is of the second type, then the further guidelines and restrictions do not apply. All plugins, be they block-only or not, are required to comply with the [Detailed Plugin Guidelines](#).

Block Plugins and the Block Directory

The Block Directory contains only Block Plugins, that is to say plugins that contain only a single, independent, top-level block and a minimum of supporting code. Block plugins have the following characteristics:

1. A specific type of WordPress plugin, with the same structure including a `readme.txt` file.
2. Containing only blocks (typically a single top-level block).

3. Contain a minimum of server-side (i.e. PHP) code.
4. Self-contained with no UI outside of the post editor.

In addition to the guidelines that apply to all WordPress plugins, Block Plugins that are submitted to the Block Directory must adhere to these guidelines:

Guidelines

1. Block Plugins are for the Block Editor.

A Block Plugin must contain a block, and a minimum of other supporting code. It may not contain any UX outside of the editor, such as WordPress options or `wp-admin` menus. Server-side code should be kept to a minimum.

Plugins that only extend or provide styles for other, pre-existing blocks are currently not eligible for inclusion in the Block Directory. At this time, they are not supported by the Block Editor's seamless install process. Only Block Plugins that register a new block are currently supported.

2. Block Plugins are separate blocks.

Block plugins are intended to be single purpose, separate, independent blocks, not bundles or compendiums of many blocks. In most cases, a Block Plugin should contain only one single top-level block. The Block Directory will not include collections of blocks that could be reasonably split up into separate, independent, blocks.

Block plugins may contain more than one block where a clearly necessary parent/child or container/content dependency exists; for example a list block that contains list item blocks.

3. Plugin and block names should reflect the block's purpose.

Plugin titles and block titles should describe what the block does in a way that helps users easily understand its purpose. In most cases the plugin title and the block title should be identical or very similar. Some examples of good plugin and block names include:

`Rainbow Block` `Sepia Image Grid` `Business Hours Block`

Please avoid plugin and block titles unrelated to the block itself, or that cannot be easily distinguished from core blocks. Some examples of unhelpful plugin and block names are things such as:

`PluginCo Inc Block` `Widget` `Image Block`

The same trademark restrictions for plugin titles exist for block titles and names. This means that a block may not be named “Facerange Block” unless developed by a legal representative of Facerange.

3a. Block names should be unique and properly namespaced.

The block name (meaning the `name` parameter to `registerBlockType()` and `name` in `block.json`) must be unique to the block. As with titles, please respect trademarks and other projects’ commonly used names, so as not to conflict with them.

The namespace prefix to the block name should reflect either the plugin author, or the plugin slug. For example:

```
name: "my-rainbow-block-plugin/rainbow", or name: "john-doe/rainbow", or name: "pluginco/rainbow".
```

The namespace may not be a reserved one such as `core` or `wordpress`.

4. Block Plugins must include a `block.json` file.

The Block Registration RFC outlines the format of a `block.json` file:

<https://github.com/WordPress/gutenberg/blob/master/docs/designers-developers/developers/block-api/block-metadata.md>

Block Plugins must include a valid `block.json` file. In addition to the requirements specified in the RFC, the `block.json` must include the following attributes:

- `name`
- `title`
- At least one of: `script`, `editorScript`
- At least one of: `style`, `editorStyle`

5. Block Plugins must work independently.

Block Plugins must function by themselves without requiring any external dependencies such as another WordPress plugin or theme.

A Block Plugin may make use of code or hooks in another WordPress plugin or theme, provided the Block Plugin is still usable without that plugin or theme installed. For example, a Recipe Block Plugin is permitted to apply a filter implemented in a slider plugin to improve its image display, as long as the Recipe Block Plugin is still usable without the slider plugin.

6. Block Plugins should work seamlessly.

Block Plugins are intended to work seamlessly and instantly when installed from the editor. That means they should not encumber the user with additional steps or prerequisites such as installing another plugin or theme, signing up for an account, or logging in or manually connecting to an external service.

No form of payment is permitted for the use of a Block Plugin. While it is allowed to use the donation link feature in the plugin's readme, or to link from the full plugin listing, the Block Plugin itself must be free to use. Block plugins that do require a paid service or include upselling and premium features are still permitted in the main WordPress Plugin Directory, subject to its guidelines.

Block Plugins may use an external/cloud API where necessary, provided it can be done seamlessly without requiring a login or activation key.

They should not rely on an external API or cloud service for functions that could be performed locally.

7. Server-side code should be kept to a minimum.

Since Block Plugins are WordPress plugins, they necessarily contain PHP code for metadata and initialization. As much as possible, they should avoid including additional PHP code or server-side libraries. The WordPress REST API should be the preferred interface to WordPress, rather than custom server-side code.

There are limits to the REST API, and situations where server-side PHP is the only performant way to implement a feature. In those situations, PHP code may be included, provided it is clearly written, used as little as possible, and well documented.

8. Must not include advertisements or promotional notices.

Block Plugins must not include code that displays alerts, dashboard notifications, or similar obtrusive messages unrelated to the purpose of the block.

👤 #2
👤 2026-10-26 07:37:07 UTC
👤 2026-10-26 07:37:07 UTC