WordPress.org Plugin Compliance Checklist

Version: 2.0

Last Updated: October 4, 2025

Purpose: A comprehensive checklist for developers and AI agents to ensure 100% WordPress.org plugin compliance

This checklist consolidates requirements from the official WordPress.org Plugin Directory Guidelines, the WordPress Plugin Check tool, PHPCS rulesets, and community best practices. Following these requirements will significantly increase the chances of first-time approval.

Table of Contents

---

1. File Structure & Initial Setup

1.1 Main Plugin File

✓	Requirement	Details
[ ]	**Single main PHP file**	Must be located in its own folder (e.g., `/my-plugin/my-plugin.php`)
[ ]	**Standard plugin header**	Must contain at minimum: `Plugin Name`, `Version`, `License`, `License URI`, `Text Domain`
[ ]	**Unique plugin name**	Must not infringe trademarks; cannot start with "WordPress" unless authorized
[ ]	**GPL-compatible license**	Strongly recommended: `GPLv2 or later`
[ ]	**Text Domain matches slug**	Text Domain must exactly match the plugin folder/slug name
[ ]	**Requires at least**	Specify minimum WordPress version (e.g., `5.0`)
[ ]	**Requires PHP**	Specify minimum PHP version (e.g., `7.2`)

Example Header:

PHP

```php
/**
 * Plugin Name: My Great Plugin
 * Description: A brief description of what my plugin does.
 * Version: 1.0.0
 * Author: Your Name
 * License: GPLv2 or later
 * License URI: https://www.gnu.org/licenses/gpl-2.0.html
 * Text Domain: my-great-plugin
 * Requires at least: 5.0
 * Requires PHP: 7.2
 */
```

1.2 Unique Naming & Prefixes

✓	Requirement	Details
[ ]	**Unique function names**	Prefix all functions with a unique identifier (3+ cha e.g., `mgp_` for "My Great Plugin")
[ ]	**Unique class names**	Prefix all classes (e.g., `class MGP_Admin`)
[ ]	**Unique constants**	Prefix all constants (e.g., `MGP_VERSION`)
[ ]	**Unique global variables**	Prefix all globals (e.g., `$mgp_settings`)
[ ]	**Avoid reserved prefixes**	Do NOT use `wp_`, `_`, `__` (reserved for WordPress
[ ]	**No function_exists() workaround**	Use truly unique names instead of wrapping in `if (function_exists())`

1.3 File Organization

✓	Requirement	Details
[ ]	**Logical folder structure**	Use sub-folders: `/includes`, `/assets`, `/templates`, `/languages`
[ ]	**Dynamic path references**	Use `plugin_dir_path(__FILE__)` for file paths
[ ]	**Dynamic URL references**	Use `plugin_dir_url(__FILE__)` or `plugins_url()` for URLs
[ ]	**No hardcoded paths**	Never hardcode `/wp-content/plugins/` paths
[ ]	**No hidden files**	Remove `.git`, `.svn`, `.DS_Store`, etc. before submission
[ ]	**No compressed files**	Remove `.zip`, `.tar.gz` files
[ ]	**No VCS directories**	Remove `.git/`, `.svn/` directories
[ ]	**No dev files**	Remove `node_modules/`, `composer.json`, `package.json` (or document them)

1.4 Security Guards

✓	Requirement	Details
[ ]	**ABSPATH check**	Add to ALL PHP files: `if ( ! defined( 'ABSPATH' ) ) { exit; }`
[ ]	**No direct file access**	Prevent direct URL access to plugin PHP files
[ ]	**No wp-load.php calls**	Never include `wp-load.php` or `wp-config.php`
[ ]	**No core file includes**	Use WordPress hooks instead of manually bootstrapping

1.5 Activation/Deactivation/Uninstall

✓	Requirement	Details
[ ]	**Activation hook**	Use `register_activation_hook(__FILE__, 'callback')` for setup tasks
[ ]	**Deactivation hook**	Use `register_deactivation_hook(__FILE__, 'callback')` for cleanup
[ ]	**Uninstall cleanup**	Implement `uninstall.php` or `register_uninstall_hook()` to remove data
[ ]	**Capability checks**	Check `current_user_can()` in activation/deactivation callbacks
[ ]	**Flush rewrite rules**	If adding custom post types/taxonomies, flush rules on activation

---

2. Plugin Header & Metadata

2.1 Required Header Fields

✓	Field	Requirement
[ ]	**Plugin Name**	Required. Must be unique and not infringe trademarks
[ ]	**Version**	Required. Must match `readme.txt` Stable tag
[ ]	**License**	Required. Must be `GPLv2 or later` (or GPL-compatible)
[ ]	**License URI**	Required. Must link to license text
[ ]	**Text Domain**	Required. Must exactly match plugin slug
[ ]	**Description**	Recommended. Brief description of functionality
[ ]	**Author**	Recommended. Your name or company
[ ]	**Author URI**	Optional. Link to your website
[ ]	**Requires at least**	Recommended. Minimum WordPress version
[ ]	**Requires PHP**	Recommended. Minimum PHP version
[ ]	**Domain Path**	Optional. If translations in subfolder (e.g., `/languages`)

2.2 Version Consistency

✓	Requirement	Details
[ ]	**Header version matches readme**	Version in main file header must match `Stable tag` in `readme.txt`
[ ]	**Increment for each release**	Version must increase with each new release (e.g., 1.0.1)
[ ]	**Semantic versioning**	Follow `MAJOR.MINOR.PATCH` format

---

3. Security: Input Sanitization

Rule: Sanitize ALL input data as soon as it's received.

3.1 Sanitization Functions

✓	Input Type	Function to Use
[ ]	**Text fields**	`sanitize_text_field()`
[ ]	**Textarea**	`sanitize_textarea_field()`
[ ]	**Email**	`sanitize_email()`
[ ]	**URL**	`esc_url_raw()`
[ ]	**File name**	`sanitize_file_name()`
[ ]	**HTML class**	`sanitize_html_class()`
[ ]	**Key (slug)**	`sanitize_key()`
[ ]	**Title**	`sanitize_title()`
[ ]	**Integer**	`intval()` or `absint()`
[ ]	**Float**	`floatval()`
[ ]	**Boolean**	`(bool)` or `rest_sanitize_boolean()`
[ ]	**Array**	`array_map()` with appropriate sanitization function
[ ]	**HTML content**	`wp_kses()` or `wp_kses_post()`

3.2 Input Sources to Sanitize

✓	Source	Example
[ ]	**$_POST**	`$name = sanitize_text_field( $_POST['name'] );`
[ ]	**$_GET**	`$id = absint( $_GET['id'] );`
[ ]	**$_REQUEST**	`$action = sanitize_key( $_REQUEST['action'] );`
[ ]	**$_COOKIE**	`$value = sanitize_text_field( $_COOKIE['key'] );`
[ ]	**$_SERVER**	`$ip = sanitize_text_field( $_SERVER['REMOTE_ADDR'] );`
[ ]	**User input**	Any data from forms, AJAX, REST API

3.3 Validation

✓	Requirement	Details
[ ]	**Validate data types**	Check if integer is actually integer, email is valid, etc.
[ ]	**Check expected values**	For dropdowns/selects, verify value is in allowed list
[ ]	**Reject invalid data**	Return error if data doesn't meet requirements

Example:

PHP

```php
// Sanitize and validate
$user_id = isset( $_POST['user_id'] ) ? absint( $_POST['user_id'] ) : 0;
if ( $user_id <= 0 ) {
    wp_die( 'Invalid user ID' );
}
```

---

4. Security: Output Escaping

Rule: Escape ALL output before sending to the browser.

4.1 Escaping Functions

✓	Output Context	Function to Use
[ ]	**HTML content**	`esc_html()`
[ ]	**HTML attributes**	`esc_attr()`
[ ]	**URLs**	`esc_url()`
[ ]	**JavaScript**	`esc_js()`
[ ]	**Textarea**	`esc_textarea()`
[ ]	**SQL**	`esc_sql()` (use `$wpdb->prepare()` instead)
[ ]	**HTML with allowed tags**	`wp_kses()` or `wp_kses_post()`

4.2 Translation + Escaping (Combined Functions)

✓	Function	Use Case
[ ]	**esc_html__()**	Translate and escape for HTML content (returns string)
[ ]	**esc_html_e()**	Translate and escape for HTML content (echoes)
[ ]	**esc_attr__()**	Translate and escape for HTML attributes (returns string)
[ ]	**esc_attr_e()**	Translate and escape for HTML attributes (echoes)
[ ]	**esc_html_x()**	Translate with context and escape for HTML
[ ]	**esc_attr_x()**	Translate with context and escape for attributes

4.3 Output Locations to Escape

✓	Location	Example
[ ]	**Echo statements**	`echo esc_html( $variable );`
[ ]	**HTML attributes**	`<input value="<?php echo esc_attr( $value ); ?>">`
[ ]	**URLs**	`<a href="<?php echo esc_url( $link ); ?>">`
[ ]	**Admin notices**	`echo '<div class="notice"><p>' . esc_html( $message ) . '</p></div>';`
[ ]	**JavaScript variables**	`var name = '<?php echo esc_js( $name ); ?>';`

Example:

PHP

```php
// Correct: Escape on output
$title = get_option( 'my_plugin_title' );
echo '<h1>' . esc_html( $title ) . '</h1>';

// Correct: Escape in attributes
$url = get_option( 'my_plugin_url' );
echo '<a href="' . esc_url( $url ) . '">Link</a>';
```

5. Security: Nonces & Capabilities

5.1 Nonces (Verify User Intent)

✓	Requirement	Details
[ ]	**Create nonce in forms**	Use `wp_nonce_field( 'action', 'nonce_name' )`
[ ]	**Verify nonce on submission**	Use `check_admin_referer( 'action', 'nonce_name' )`
[ ]	**AJAX nonce creation**	Use `wp_create_nonce( 'action' )`
[ ]	**AJAX nonce verification**	Use `check_ajax_referer( 'action', 'nonce_name' )`
[ ]	**URL nonce creation**	Use `wp_nonce_url( $url, 'action' )`
[ ]	**URL nonce verification**	Use `wp_verify_nonce( $_GET['_wpnonce'], 'action' )`

Example:

PHP

```php
// Form with nonce
<form method="post">
    <?php wp_nonce_field( 'my_action', 'my_nonce' ); ?>
    <input type="text" name="my_field">
    <input type="submit">
</form>

// Verify nonce on submission
if ( isset( $_POST['my_nonce'] ) && wp_verify_nonce( $_POST['my_nonce'],
'my_action' ) ) {
    // Process form
}
```

5.2 User Capabilities

✓	Requirement	Details
[ ]	**Check capabilities**	Use `current_user_can( 'capability' )` before any action
[ ]	**Admin actions**	Require `manage_options` or appropriate capability
[ ]	**Content editing**	Require `edit_posts`, `edit_pages`, etc.
[ ]	**File uploads**	Require `upload_files`
[ ]	**Plugin management**	Require `activate_plugins`

Common Capabilities:

- `manage_options` - Administrator
- `edit_posts` - Editor, Author, Contributor
- `edit_pages` - Editor
- `publish_posts` - Editor, Author
- `upload_files` - Editor, Author
- `read` - Subscriber

Example:

PHP

```php
// Check capability before processing
if ( ! current_user_can( 'manage_options' ) ) {
    wp_die( 'You do not have sufficient permissions.' );
}
```

5.3 Combined Nonce + Capability Check

✓	Requirement	Details
[ ]	**Always use both**	Check nonce AND capability for all actions
[ ]	**Check before processing**	Verify before database writes, file operations, etc.

Example:

PHP

```php
// Complete security check
if ( ! isset( $_POST['my_nonce'] ) || ! wp_verify_nonce( $_POST['my_nonce'], 'my_action' ) ) {
    wp_die( 'Security check failed' );
}

if ( ! current_user_can( 'manage_options' ) ) {
    wp_die( 'Insufficient permissions' );
}

// Now safe to process
$value = sanitize_text_field( $_POST['my_field'] );
update_option( 'my_option', $value );
```

6. Database Queries

6.1 Using $wpdb->prepare()

✓	Requirement	Details
[ ]	**Always use prepare()**	For ALL queries with variable data
[ ]	**Use placeholders**	`%s` for strings, `%d` for integers, `%f` for floats
[ ]	**No direct concatenation**	Never concatenate variables into SQL strings
[ ]	**Prepare before execute**	Call `prepare()` before `query()`, `get_results()`, etc.

Example:

PHP

```php
// WRONG: Direct concatenation (SQL injection vulnerability)
$user_id = $_GET['user_id'];
$results = $wpdb->get_results( "SELECT * FROM {$wpdb->prefix}my_table WHERE user_id = $user_id" );

// CORRECT: Using prepare()
$user_id = absint( $_GET['user_id'] );
$results = $wpdb->get_results(
    $wpdb->prepare(
        "SELECT * FROM {$wpdb->prefix}my_table WHERE user_id = %d",
```

```
        $user_id
    )
);
```

6.2 Database Operations

✓	Requirement	Details
[ ]	**Use WordPress APIs first**	Prefer `get_posts()` , `WP_Query` , `get_option()` , etc.
[ ]	**Avoid direct queries**	Only use `$wpdb` when necessary
[ ]	**Use $wpdb methods**	`get_results()` , `get_row()` , `get_var()` , `insert()` , `update()` , `delete()`
[ ]	**Table prefix**	Always use `$wpdb->prefix` or `$wpdb->posts` , `$wpdb->users` , etc.
[ ]	**Error handling**	Check for errors with `$wpdb->last_error`

6.3 Restricted Database Functions

✓	Requirement	Details
[ ]	**No PHP database extensions**	Don't use `mysqli_*` , `PDO` , `pg_*` , etc.
[ ]	**Use WordPress abstraction**	Always use `$wpdb` class
[ ]	**No direct MySQL**	Don't connect to MySQL directly

---

7. WordPress APIs & Libraries

7.1 Use WordPress Functions

✓	Instead of...	Use WordPress Function
[ ]	`file_get_contents()` (remote)	`wp_remote_get()`
[ ]	`curl_*()`	`wp_remote_get()` , `wp_remote_post()`
[ ]	`fopen()` (remote)	`wp_remote_get()`
[ ]	`mail()`	`wp_mail()`
[ ]	`$_COOKIE` (direct)	`wp_set_auth_cookie()` , `wp_clear_auth_cookie()`
[ ]	`header()` (redirect)	`wp_redirect()` or `wp_safe_redirect()`
[ ]	`json_encode()`	Acceptable (not required to use WP function)
[ ]	`file_get_contents()` (local)	Acceptable for local files
[ ]	`file_put_contents()` (local)	Acceptable for local files

7.2 Use WordPress Libraries

✓	Requirement	Details
[ ]	**jQuery**	Use WordPress's bundled jQuery (`jquery` handle)
[ ]	**jQuery UI**	Use WordPress's bundled jQuery UI components
[ ]	**Underscore.js**	Use WordPress's bundled Underscore (`underscore` handle)
[ ]	**Backbone.js**	Use WordPress's bundled Backbone (`backbone` handle)
[ ]	**Moment.js**	Use WordPress's bundled Moment (`moment` handle)
[ ]	**React**	Use WordPress's bundled React (if available)
[ ]	**No duplicate libraries**	Never include your own copy of these libraries

7.3 HTTP API

✓	Requirement	Details
[ ]	**wp_remote_get()**	For GET requests
[ ]	**wp_remote_post()**	For POST requests
[ ]	**wp_remote_request()**	For custom requests
[ ]	**Check for errors**	Use `is_wp_error()`
[ ]	**Set timeout**	Use `timeout` parameter (default 5 seconds)
[ ]	**Handle response**	Use `wp_remote_retrieve_body()`, `wp_remote_retrieve_response_code()`

Example:

PHP

```php
$response = wp_remote_get( 'https://api.example.com/data', array(
    'timeout' => 10,
) );

if ( is_wp_error( $response ) ) {
    // Handle error
    $error_message = $response->get_error_message();
} else {
    $body = wp_remote_retrieve_body( $response );
    $data = json_decode( $body );
}
```

---

8. Scripts & Styles (Asset Enqueuing)

8.1 Enqueuing Scripts

✓	Requirement	Details
[ ]	**Use wp_enqueue_script()**	Never hardcode `<script>` tags
[ ]	**Register first (optional)**	Use `wp_register_script()` if needed
[ ]	**Declare dependencies**	Specify dependencies array (e.g., `array('jquery')`)
[ ]	**Version number**	Use plugin version or file modification time
[ ]	**Load in footer**	Set `$in_footer` parameter to `true`
[ ]	**Conditional loading**	Only enqueue on pages where needed
[ ]	**Use handles**	Don't duplicate WordPress core script handles

Example:

PHP

```php
function my_plugin_enqueue_scripts() {
    // Only on plugin's admin page
    if ( isset( $_GET['page'] ) && $_GET['page'] === 'my-plugin' ) {
        wp_enqueue_script(
            'my-plugin-admin',                          // Handle
            plugins_url( 'assets/js/admin.js', __FILE__ ), // Source
            array( 'jquery' ),                          // Dependencies
            '1.0.0',                                    // Version
            true                                        // In footer
        );
    }
}
add_action( 'admin_enqueue_scripts', 'my_plugin_enqueue_scripts' );
```

8.2 Enqueuing Styles

✓	Requirement	Details
[ ]	**Use wp_enqueue_style()**	Never hardcode `<link>` tags
[ ]	**Register first (optional)**	Use `wp_register_style()` if needed
[ ]	**Declare dependencies**	Specify dependencies array if needed
[ ]	**Version number**	Use plugin version or file modification time
[ ]	**Media type**	Specify media type (e.g., `all`, `screen`, `print`)
[ ]	**Conditional loading**	Only enqueue on pages where needed

Example:

PHP

```php
function my_plugin_enqueue_styles() {
    wp_enqueue_style(
        'my-plugin-style',
        plugins_url( 'assets/css/style.css', __FILE__ ),
        array(),
        '1.0.0',
        'all'
    );
}
add_action( 'wp_enqueue_scripts', 'my_plugin_enqueue_styles' );
```

8.3 Inline Scripts & Styles

✓	Requirement	Details
[ ]	**Use wp_add_inline_script()**	For small inline scripts tied to enqueued handle
[ ]	**Use wp_add_inline_style()**	For small inline styles tied to enqueued handle
[ ]	**Localize scripts**	Use `wp_localize_script()` to pass PHP data to JavaScript

Example:

PHP

```php
// Enqueue script
wp_enqueue_script( 'my-plugin-script', plugins_url( 'js/script.js', __FILE__
), array(), '1.0.0', true );

// Add inline script
wp_add_inline_script( 'my-plugin-script', 'console.log("Plugin loaded");' );

// Localize script (pass data to JS)
wp_localize_script( 'my-plugin-script', 'myPluginData', array(
    'ajax_url' => admin_url( 'admin-ajax.php' ),
    'nonce'    => wp_create_nonce( 'my_action' ),
) );
```

8.4 Asset Loading Strategy

✓	Requirement	Details
[ ]	**Set loading strategy**	Use `defer` or `async` for non-critical scripts
[ ]	**Use wp_script_add_data()**	Set strategy: `wp_script_add_data( 'handle', 'strategy', 'defer' )`
[ ]	**Load in footer**	Prefer footer loading for better performance

---

9. Internationalization (i18n)

9.1 Text Domain

✓	Requirement	Details
[ ]	**Text Domain in header**	Must be present and match plugin slug exactly
[ ]	**Domain Path (if needed)**	If translations in subfolder: `Domain Path: /languages`
[ ]	**Load text domain**	Use `load_plugin_textdomain()`

Example:

PHP

```
function my_plugin_load_textdomain() {
    load_plugin_textdomain( 'my-great-plugin', false, dirname(
plugin_basename( __FILE__ ) ) . '/languages' );
}
add_action( 'plugins_loaded', 'my_plugin_load_textdomain' );
```

9.2 Translation Functions

✓	Function	Use Case
[ ]	**__()**	Translate string (returns)
[ ]	**_e()**	Translate and echo string
[ ]	**_x()**	Translate with context
[ ]	**_ex()**	Translate with context and echo
[ ]	**_n()**	Translate singular/plural
[ ]	**_nx()**	Translate singular/plural with context
[ ]	**esc_html__()**	Translate and escape for HTML (returns)
[ ]	**esc_html_e()**	Translate and escape for HTML (echoes)
[ ]	**esc_attr__()**	Translate and escape for attributes (returns)
[ ]	**esc_attr_e()**	Translate and escape for attributes (echoes)

9.3 Translation Best Practices

✓	Requirement	Details
[ ]	**Wrap all strings**	All user-facing strings must be translatable
[ ]	**Use text domain**	Always pass text domain as second parameter
[ ]	**No variables in strings**	Don't use variables inside translation functions
[ ]	**Use placeholders**	Use `sprintf()` for dynamic content
[ ]	**Provide context**	Use `_x()` when string could be ambiguous

Example:

```php
// WRONG: Variable inside translation
echo __( "Hello $name", 'my-plugin' );

// CORRECT: Use sprintf with placeholder
echo sprintf( __( 'Hello %s', 'my-plugin' ), $name );

// CORRECT: With context
echo _x( 'Post', 'noun', 'my-plugin' ); // vs. 'Post' as verb
```

---

10. PHP Coding Standards

10.1 PHP Version & Syntax

✓	Requirement	Details
[ ]	**Declare minimum PHP**	In plugin header: `Requires PHP: 7.2` (or higher)
[ ]	**Full PHP tags**	Always use `<?php ?>`, never short tags `<? ?>`
[ ]	**No alternative tags**	Don't use `<% %>` or `<script language="php">`
[ ]	**No closing tag**	Omit `?>` at end of PHP-only files
[ ]	**UTF-8 encoding**	Save all files as UTF-8 without BOM
[ ]	**No BOM**	Byte Order Mark is not allowed

10.2 Code Readability

✓	Requirement	Details
[ ]	**Human-readable code**	No obfuscation, minification, or packing of PHP
[ ]	**Meaningful names**	Use descriptive function/variable names
[ ]	**No unclear naming**	Avoid names like `$z12sdf813d`
[ ]	**Include source**	If JS/CSS is minified, include unminified source or link to repository
[ ]	**Document build tools**	If using build tools, document how to use them

10.3 Error Handling

✓	Requirement	Details
[ ]	**No error_reporting()**	Don't change PHP error reporting settings
[ ]	**No ini_set() for errors**	Don't use `ini_set('display_errors')`
[ ]	**Use error_log()**	For logging errors
[ ]	**No var_dump()**	Remove all debugging statements
[ ]	**No print_r()**	Remove all debugging statements
[ ]	**Handle exceptions**	Use try/catch for exception-prone code

10.4 Global State

✓	Requirement	Details
[ ]	**No timezone changes**	Don't use `date_default_timezone_set()`
[ ]	**No global overrides**	Don't modify `$wpdb` or other core globals
[ ]	**Self-contained**	Plugin should not alter global WordPress environment

---

11. Forbidden & Discouraged Functions

11.1 Absolutely Forbidden (Severity 7 - Rejection)

✓	Function/Construct	Reason
[ ]	**eval()**	Security risk - arbitrary code execution
[ ]	**create_function()**	Deprecated and security risk
[ ]	**goto**	Poor code structure
[ ]	**Backtick operator**	Shell command execution risk
[ ]	**HEREDOC**	Not allowed
[ ]	**NOWDOC**	Not allowed
[ ]	**base64_decode()** (for obfuscation)	Code obfuscation
[ ]	**str_rot13()**	Code obfuscation
[ ]	**move_uploaded_file()**	Security risk
[ ]	**passthru()**	Shell command execution
[ ]	**proc_open()**	Shell command execution
[ ]	**exec()**	Shell command execution (use with extreme cauti
[ ]	**system()**	Shell command execution
[ ]	**shell_exec()**	Shell command execution

11.2 Forbidden WordPress Internal Functions

✓	Function	Reason
[ ]	**_cleanup_header_comment()**	Internal function
[ ]	**_get_plugin_data_markup_translate()**	Internal function
[ ]	**_transition_post_status()**	Internal function
[ ]	**_wp_post_revision_fields()**	Internal function
[ ]	**do_shortcode_tag()**	Internal function
[ ]	**get_post_type_labels()**	Internal function
[ ]	**wp_get_sidebars_widgets()**	Internal function
[ ]	**wp_get_widget_defaults()**	Internal function

11.3 Discouraged Functions

✓	Function	Alternative
[ ]	**set_time_limit()**	Avoid or use temporarily
[ ]	**ini_set()**	Avoid changing PHP settings
[ ]	**ini_alter()**	Avoid changing PHP settings
[ ]	**dl()**	Don't load PHP extensions
[ ]	**error_reporting()**	Don't change error reporting
[ ]	**extract()**	Use explicit variable assignment

11.4 Deprecated WordPress Functions

✓	Requirement	Details
[ ]	**No deprecated functions**	Check WordPress Codex for deprecated functions
[ ]	**No deprecated classes**	Use current WordPress classes
[ ]	**No deprecated parameters**	Check function signatures
[ ]	**No deprecated constants**	Use current constants

---

12. Performance Optimization

12.1 Script & Style Performance

✓	Requirement	Details
[ ]	**Scripts in footer**	Load non-critical scripts in footer
[ ]	**Conditional loading**	Only load assets on pages where needed
[ ]	**Minimize file size**	Keep cumulative scripts < 293 KB
[ ]	**Minimize stylesheet size**	Keep cumulative styles < 293 KB
[ ]	**No global loading**	Don't load assets on ALL pages unless necessary
[ ]	**Use loading strategy**	Implement `defer` or `async` for scripts
[ ]	**Combine files**	Reduce number of HTTP requests

12.2 Database Performance

✓	Requirement	Details
[ ]	**Avoid slow queries**	Don't use `posts_per_page => -1` without good reason
[ ]	**Use pagination**	Limit query results
[ ]	**Index custom tables**	Add indexes to frequently queried columns
[ ]	**Cache results**	Use transients for expensive queries
[ ]	**Avoid meta queries**	Meta queries are slow; use custom tables if needed

12.3 General Performance

✓	Requirement	Details
[ ]	**Use transients**	Cache expensive operations with `set_transient()`
[ ]	**Use object cache**	Use `wp_cache_set()` for frequently accessed data
[ ]	**Minimize external calls**	Limit HTTP requests to external services
[ ]	**Set timeouts**	Use short timeouts for external requests (5-10 seconds)
[ ]	**Lazy load**	Load resources only when needed
[ ]	**Use WP Cron**	Schedule heavy tasks with `wp_schedule_event()`

12.4 Image Optimization

✓	Requirement	Details
[ ]	**Use WordPress functions**	Use `wp_get_attachment_image()` for images
[ ]	**Responsive images**	WordPress generates srcset automatically
[ ]	**Optimize file size**	Compress images before including

---

13. WordPress.org Guidelines (18 Rules)

Guideline 1: GPL Compatibility

✓	Requirement	Details
[ ]	**GPL-compatible license**	All code, data, images must be GPLv2 or later compatible
[ ]	**Third-party libraries**	All included libraries must be GPL-compatible
[ ]	**License file**	Include `LICENSE.txt` or reference in header
[ ]	**Check library licenses**	MIT, BSD, Apache 2.0 are compatible; proprietary is not

Guideline 2: Developer Responsibility

✓	Requirement	Details
[ ]	**Verify all files**	Ensure all files comply with guidelines
[ ]	**Check third-party code**	Verify licensing and terms of use
[ ]	**No circumvention**	Don't intentionally write code to bypass guidelines
[ ]	**Maintain compliance**	Don't restore removed code

Guideline 3: Stable Version Available

✓	Requirement	Details
[ ]	**Keep WordPress.org updated**	Don't distribute elsewhere without updating repo
[ ]	**Stable version in directory**	Users download from WordPress.org, not dev environn

Guideline 4: Human-Readable Code

✓	Requirement	Details
[ ]	**No obfuscation**	No p,a,c,k,e,r, uglify mangle, etc.
[ ]	**Clear naming**	Use meaningful variable/function names
[ ]	**Provide source**	Include unminified source or link to repository
[ ]	**Document build tools**	Explain how to use development tools

Guideline 5: No Trialware

✓	Requirement	Details
[ ]	**No locked features**	All code in plugin must be fully functional
[ ]	**No trial periods**	No time-limited functionality
[ ]	**No quotas**	No usage limits that lock features
[ ]	**No sandbox-only APIs**	Must provide real functionality
[ ]	**Upselling allowed**	Can promote paid add-ons (see Guideline 11)

Guideline 6: Software as a Service (SaaS) Permitted

✓	Requirement	Details
[ ]	**Document service**	Clearly explain external service in readme
[ ]	**Link to terms**	Provide link to service's Terms of Use
[ ]	**Substantial functionality**	Service must provide real value
[ ]	**No license validation only**	Service can't exist solely to validate licenses
[ ]	**No fake services**	Don't move code to external service to fake functionality
[ ]	**No storefronts**	Plugin can't be just a front-end for external products

Guideline 7: No Tracking Without Consent

✓	Requirement	Details
[ ]	**Opt-in required**	User must explicitly consent to tracking
[ ]	**Document data collection**	Explain what data is collected and why
[ ]	**Privacy policy**	Provide clear privacy policy
[ ]	**No automated collection**	Don't collect data without confirmation
[ ]	**No misleading consent**	Don't trick users into submitting data
[ ]	**No offloading assets**	Don't load images/scripts from external servers for trac
[ ]	**No undocumented blocklists**	Document use of external data
[ ]	**No ad tracking**	No third-party ad mechanisms that track users

Exception: SaaS plugins may contact their service (but must document it).

Guideline 8: No Executable Code via Third-Party

✓	Requirement	Details
[ ]	**No remote code execution**	Don't download and execute code from externa
[ ]	**No eval() of remote data**	Don't eval() content from APIs
[ ]	**No create_function() of remote data**	Don't dynamically create functions from extern

Exception: Webhooks that trigger existing plugin functions are OK.

Guideline 9: No Illegal/Dishonest/Offensive Actions

✓	Requirement	Details
[ ]	**Legal compliance**	Follow all applicable laws
[ ]	**No spam**	Don't send unsolicited emails
[ ]	**No malware**	No malicious code
[ ]	**No deceptive practices**	Be honest about functionality
[ ]	**No circumvention**	Don't help users break guidelines

Guideline 10: No External Links Without Permission

✓	Requirement	Details
[ ]	**No auto-inserted links**	Don't add links to user's public site without permission
[ ]	**No hidden credits**	Don't hide attribution links in content
[ ]	**Ask permission**	Use opt-in setting for any external links

Guideline 11: Don't Hijack Admin Dashboard

✓	Requirement	Details
[ ]	**Minimal admin notices**	Don't show constant nags
[ ]	**Dismissible notices**	Make notices dismissible
[ ]	**Limit promotions**	Show upsells on plugin pages only
[ ]	**No large banners**	Don't take over admin with ads
[ ]	**Respect user experience**	Don't be intrusive

Guideline 12: No Spam in Readme

✓	Requirement	Details
[ ]	**No keyword stuffing**	Write naturally
[ ]	**No excessive links**	Limit promotional links
[ ]	**No misleading tags**	Use accurate tags
[ ]	**Professional content**	Write clear, helpful descriptions

Guideline 13: Use WordPress Default Libraries

✓	Requirement	Details
[ ]	**Use bundled jQuery**	Don't include your own copy
[ ]	**Use bundled libraries**	Use WordPress's Underscore, Backbone, etc.
[ ]	**No duplicate libraries**	Don't bundle libraries WordPress provides
[ ]	**Check available libraries**	See what WordPress includes before bundling

Guideline 14: Avoid Frequent Commits

✓	Requirement	Details
[ ]	**Batch changes**	Don't commit every small change
[ ]	**Test before committing**	Ensure changes work
[ ]	**Meaningful commits**	Each commit should be a logical unit

Guideline 15: Increment Version Numbers

✓	Requirement	Details
[ ]	**Increase version**	Each release must have higher version number
[ ]	**Update both files**	Change version in main file AND readme.txt
[ ]	**Semantic versioning**	Use MAJOR.MINOR.PATCH format
[ ]	**Match stable tag**	Stable tag must match actual tagged version

Guideline 16: Complete Plugin at Submission

✓	Requirement	Details
[ ]	**Functional plugin**	Must be fully working at submission
[ ]	**No placeholders**	Don't submit incomplete code
[ ]	**No "coming soon"**	All advertised features must exist
[ ]	**Test thoroughly**	Ensure plugin works before submitting

Guideline 17: Respect Trademarks

✓	Requirement	Details
[ ]	**No trademark infringement**	Don't use others' trademarks in name/slug
[ ]	**No brand confusion**	Don't imply official affiliation
[ ]	**Check name availability**	Search for existing plugins with similar names
[ ]	**Rename if needed**	Be prepared to change name if trademark issue

Guideline 18: WordPress.org's Right to Maintain Directory

✓	Requirement	Details
[ ]	**Accept decisions**	WordPress.org has final say
[ ]	**Respond to requests**	Reply to plugin team emails
[ ]	**Fix issues promptly**	Address security issues immediately
[ ]	**Follow new guidelines**	Comply with updated requirements

---

14. readme.txt Requirements

14.1 Required Fields

✓	Field	Requirement
[ ]	**Plugin Name**	Must match main file header
[ ]	**Contributors**	WordPress.org usernames (comma-separated)
[ ]	**Tags**	Relevant keywords (max 12)
[ ]	**Requires at least**	Minimum WordPress version
[ ]	**Tested up to**	Latest WordPress version tested
[ ]	**Stable tag**	Current version number (must match tagged version in SVN)
[ ]	**License**	GPL-compatible license
[ ]	**License URI**	Link to license

14.2 Recommended Sections

✓	Section	Content
[ ]	**Short Description**	Brief description (max 150 characters)
[ ]	**Description**	Detailed explanation of features
[ ]	**Installation**	Step-by-step installation instructions
[ ]	**Frequently Asked Questions**	Common questions and answers
[ ]	**Screenshots**	List of screenshots with descriptions
[ ]	**Changelog**	Version history with changes
[ ]	**Upgrade Notice**	Important notes for users upgrading

14.3 Optional but Helpful

✓	Field	Purpose
[ ]	**Requires PHP**	Minimum PHP version
[ ]	**Donate link**	Link to donation page

14.4 Formatting

✓	Requirement	Details
[ ]	**Use markdown**	Format with markdown syntax
[ ]	**Headers**	Use `==` for h2, `=` for h3
[ ]	**Lists**	Use `*` or `1.` for lists
[ ]	**Code blocks**	Use backticks for code
[ ]	**No HTML**	Use markdown, not HTML tags

Example readme.txt structure:

```
Plain Text
```

```
=== Plugin Name ===
Contributors: username
Tags: tag1, tag2
Requires at least: 5.0
Tested up to: 6.4
Stable tag: 1.0.0
License: GPLv2 or later
License URI: https://www.gnu.org/licenses/gpl-2.0.html

Short description here.

== Description ==

Detailed description here.

== Installation ==

1. Upload plugin
2. Activate
3. Configure

== Frequently Asked Questions ==

= Question? =

Answer.

== Changelog ==

= 1.0.0 =
* Initial release
```

---

15. SVN & Submission Process

15.1 Before Submission

✓	Requirement	Details
[ ]	**Run Plugin Check tool**	Install and run official Plugin Check plugin
[ ]	**Fix all errors**	Address all errors reported by Plugin Check
[ ]	**Review warnings**	Consider fixing warnings
[ ]	**Test thoroughly**	Test in clean WordPress install
[ ]	**Enable 2FA**	Two-Factor Authentication required on WordPress.org account
[ ]	**Clean package**	Remove dev files, VCS directories, etc.
[ ]	**Verify versions**	Ensure version numbers match everywhere

15.2 Submission

✓	Requirement	Details
[ ]	**Create ZIP file**	Package plugin folder as .zip
[ ]	**Submit via WordPress.org**	Go to https://wordpress.org/plugins/developers/add/
[ ]	**Wait for auto-scan**	Plugin Check tool runs automatically
[ ]	**Fix auto-scan errors**	Address any errors before human review
[ ]	**Wait for human review**	Can take days to weeks
[ ]	**Respond to feedback**	Reply to reviewer emails promptly

15.3 SVN Structure

✓	Directory	Purpose
[ ]	**/trunk/**	Development version (latest code)
[ ]	**/tags/**	Released versions (e.g., `/tags/1.0.0/` , `/tags/1.0.1/`)
[ ]	**/branches/**	Optional: experimental branches
[ ]	**/assets/**	Screenshots, banners, icons for WordPress.org

15.4 Initial SVN Commit

✓	Step	Command
[ ]	**Checkout SVN**	`svn co https://plugins.svn.wordpress.org/your-plugin-slug`
[ ]	**Copy files to trunk**	Copy all plugin files to `/trunk/`
[ ]	**Add files**	`svn add trunk/*`
[ ]	**Commit trunk**	`svn ci -m "Initial commit"`
[ ]	**Create tag**	`svn cp trunk tags/1.0.0`
[ ]	**Commit tag**	`svn ci -m "Tagging version 1.0.0"`
[ ]	**Verify stable tag**	Ensure `readme.txt` has `Stable tag: 1.0.0`

15.5 Updating Plugin

✓	Step	Details
[ ]	**Update trunk**	Make changes in `/trunk/`
[ ]	**Update version**	Change version in main file and readme.txt
[ ]	**Update changelog**	Add changes to readme.txt changelog
[ ]	**Commit trunk**	`svn ci -m "Update to 1.0.1"`
[ ]	**Create new tag**	`svn cp trunk tags/1.0.1`
[ ]	**Commit tag**	`svn ci -m "Tagging version 1.0.1"`
[ ]	**Update stable tag**	Change `Stable tag` in readme.txt to `1.0.1`

15.6 Assets (Screenshots, Banners, Icons)

✓	Asset	Specifications
[ ]	**Screenshots**	`screenshot-1.png`, `screenshot-2.png`, etc. (ideally 1280x720)
[ ]	**Banner**	`banner-772x250.png` (and optionally `banner-1544x500.png` for retina)
[ ]	**Icon**	`icon-128x128.png` and `icon-256x256.png`
[ ]	**Location**	Place in `/assets/` directory (not `/trunk/`)

---

16. Common Rejection Reasons

16.1 Security Issues

✓	Issue	Solution
[ ]	**Missing sanitization**	Sanitize all input with appropriate functions
[ ]	**Missing escaping**	Escape all output with `esc_html()` , `esc_attr()` , etc.
[ ]	**No nonce checks**	Add nonce verification to all forms/actions
[ ]	**No capability checks**	Check `current_user_can()` before actions
[ ]	**SQL injection**	Use `$wpdb->prepare()` for all queries
[ ]	**No ABSPATH check**	Add `if ( ! defined( 'ABSPATH' ) ) exit;` to all files

16.2 Guideline Violations

✓	Issue	Solution
[ ]	**Calling wp-load.php**	Use WordPress hooks instead
[ ]	**Hardcoded paths**	Use `plugin_dir_path()` , `plugins_url()`
[ ]	**Not using WP libraries**	Remove custom jQuery, use WordPress's bundled version
[ ]	**Tracking without consent**	Add opt-in mechanism and document in readme
[ ]	**Obfuscated code**	Provide unminified source code
[ ]	**Trademark issues**	Rename plugin to avoid trademark conflicts

16.3 Code Quality Issues

✓	Issue	Solution
[ ]	**Version mismatch**	Ensure version matches in header, readme, and SVN tag
[ ]	**Text domain mismatch**	Text domain must exactly match plugin slug
[ ]	**Using deprecated functions**	Replace with current WordPress functions
[ ]	**Short PHP tags**	Use `<?php` instead of `<?`
[ ]	**Dev files included**	Remove `node_modules/`, `.git/`, etc.

16.4 Documentation Issues

✓	Issue	Solution
[ ]	**Incomplete readme**	Fill out all required sections
[ ]	**Missing service documentation**	Document external services in readme
[ ]	**No privacy policy**	Add privacy policy if collecting data
[ ]	**Spam in readme**	Remove keyword stuffing, excessive links

---

17. Plugin Check Tool Requirements

The official Plugin Check tool (https://github.com/WordPress/plugin-check) runs 22 automated checks. Ensure your plugin passes all of them.

17.1 Plugin Repository Checks

✓	Check	What It Tests
[ ]	**i18n_usage**	Internationalization best practices
[ ]	**code_obfuscation**	Detects obfuscation tools usage
[ ]	**file_type**	Detects hidden files, compressed files, VCS directories
[ ]	**plugin_header_fields**	Validates required header fields
[ ]	**late_escaping**	Ensures output is escaped
[ ]	**plugin_updater**	Prevents custom updaters
[ ]	**plugin_review_phpcs**	Runs PHP_CodeSniffer with WordPress rules
[ ]	**direct_db_queries**	Checks for direct database queries
[ ]	**enqueued_resources**	Validates proper script/style enqueuing
[ ]	**plugin_readme**	Validates readme.txt format and content
[ ]	**localhost**	Detects localhost/127.0.0.1 references
[ ]	**no_unfiltered_uploads**	Checks for ALLOW_UNFILTERED_UPLOADS
[ ]	**trademarks**	Checks for trademark usage in slug
[ ]	**offloading_files**	Detects unnecessary remote services

17.2 Security Checks

✓	Check	What It Tests
[ ]	**late_escaping**	Output escaping
[ ]	**direct_db_queries**	SQL injection prevention

17.3 Performance Checks

✓	Check	What It Tests
[ ]	**performant_wp_query_params**	Slow WP_Query parameters
[ ]	**enqueued_scripts_in_footer**	Scripts loaded in footer
[ ]	**enqueued_scripts_size**	Total script size < 293 KB
[ ]	**enqueued_styles_size**	Total stylesheet size < 293 KB
[ ]	**enqueued_styles_scope**	Stylesheets not loaded on all pages
[ ]	**enqueued_scripts_scope**	Scripts not loaded on all pages
[ ]	**non_blocking_scripts**	Non-blocking loading strategy
[ ]	**image_functions**	Proper image insertion functions

17.4 PHPCS Rules (plugin-check.ruleset.xml)

✓	Rule Category	Key Rules
[ ]	**Database**	PreparedSQL, PreparedSQLPlaceholders, RestrictedClasses, RestrictedFunctions
[ ]	**Security**	NonceVerification, ValidatedSanitizedInput, PluginMenuSlug
[ ]	**PHP Restrictions**	No backticks, HEREDOC, goto, short tags, alternative tags, BOM
[ ]	**WordPress Best Practices**	No deprecated functions/classes, use alternative functions
[ ]	**Forbidden Functions**	No eval, create_function, passthru, proc_open, etc.

Final Pre-Submission Checklist

Critical Items

✓	Item
[ ]	All security checks pass (sanitization, escaping, nonces, capabilities)
[ ]	All database queries use `$wpdb->prepare()`
[ ]	All files have ABSPATH check
[ ]	No wp-load.php or wp-config.php includes
[ ]	Text domain matches plugin slug exactly
[ ]	Version numbers match in header, readme, and SVN tag
[ ]	GPL-compatible license declared
[ ]	No obfuscated or minified PHP code
[ ]	No forbidden functions (eval, create_function, goto, etc.)
[ ]	Scripts and styles properly enqueued
[ ]	Using WordPress bundled libraries (jQuery, etc.)
[ ]	readme.txt complete and properly formatted
[ ]	Plugin Check tool shows no errors
[ ]	2FA enabled on WordPress.org account
[ ]	All dev files removed from package
[ ]	Tested in clean WordPress install
[ ]	All 18 WordPress.org guidelines followed

References

1. David Bryan. (2025, June 18). *How to Build & Publish a Plugin to WordPress.Org by following WordPress Plugin Coding Standards*. Retrieved from attached document.
2. WordPress. (2025). *WordPress Plugin Check - GitHub Repository*. https://github.com/WordPress/plugin-check

3. WordPress. (2025). *Plugin Check Documentation - Checks*. https://github.com/WordPress/plugin-check/blob/trunk/docs/checks.md
4. WordPress. (2024, March 15). *Detailed Plugin Guidelines*. https://developer.wordpress.org/plugins/wordpress-org/detailed-plugin-guidelines/
5. WordPress. (2025). *plugin-check.ruleset.xml - PHPCS Ruleset*. https://raw.githubusercontent.com/WordPress/plugin-check/refs/heads/trunk/phpcs-rulesets/plugin-check.ruleset.xml
6. WordPress. (2025). *plugin-review.xml - PHPCS Ruleset*. https://raw.githubusercontent.com/WordPress/plugin-check/refs/heads/trunk/phpcs-rulesets/plugin-review.xml
7. WordPress. (n.d.). *Plugin Handbook*. https://developer.wordpress.org/plugins/
8. WordPress. (n.d.). *Plugin Readme File Standard*. https://developer.wordpress.org/plugins/wordpress-org/how-your-readme-txt-works/
9. WordPress. (n.d.). *How to use Subversion (SVN)*. https://developer.wordpress.org/plugins/wordpress-org/how-to-use-subversion/

Document Version: 2.0

Last Updated: October 4, 2025

Compiled by: Manus AI