Abilities API

- Core Concepts
- Goals and Benefits
- Use Cases
- Registration Example

The Abilities API is only available for WordPress 6.9 and above.

The WordPress Abilities API provides a standardized way to register and discover distinct units of functionality within a WordPress site. These units, called “Abilities”, represent specific actions or capabilities that components can perform, with clearly defined inputs, outputs, and permissions.

It acts as a central registry, making it easier for different parts of WordPress, third-party plugins, themes, and external systems (like AI agents) to understand and interact with the capabilities available on a specific site.

Core Concepts

- **Ability:** A distinct piece of functionality with a unique name following the `namespace/ability-name` pattern. Each ability has a human-readable name and description, input/output definitions (using JSON Schema), a category assignment, optional permissions, and an associated callback function for execution. Each registered Ability is an instance of the `WP_Ability` class.
- **Category:** A way to organize related abilities. Each ability must belong to exactly one category. Categories have a slug, label, and description. Each registered category is an instance of the `WP_Ability_Category` class.
- **Registry:** A central, singleton object (`WP_Abilities_Registry`) that holds all registered abilities. It provides methods for registering, unregistering, finding, and querying abilities. Similarly, `WP_Abilities_Category_Registry` manages all registered categories.
- **Callback:** The PHP function or method executed when an ability is called via `WP_Ability::execute()`.
- **Schema:** JSON Schema definitions for an ability’s expected input (`input_schema`) and its returned output (`output_schema`). This allows for validation and helps agents understand how to use the ability.

- **Permission Callback:** An optional function that determines if the current user can execute a specific ability.
- **Namespace:** The first part of an ability name (before the slash), typically matching the plugin or component name that registers the ability.

Goals and Benefits

- **Standardization:** Provides a single, consistent way to expose site capabilities.
- **Discoverability:** Makes functionality easily discoverable by AI systems and automation tools.
- **Validation:** Built-in input/output validation using JSON Schema ensures data integrity.
- **Security:** Permission callbacks provide fine-grained access control.
- **Extensibility:** Simple registration pattern allows any plugin or theme to expose their capabilities.
- **AI-Friendly:** Machine-readable format enables intelligent automation and AI agent interactions.

Use Cases

- **AI Integration:** Allow AI agents to discover and interact with site capabilities.
- **Plugin Interoperability:** Enable plugins to discover and use each other’s functionality.
- **Automation Tools:** Provide programmatic access to site features.
- **API Documentation:** Self-documenting capabilities with schema validation.
- **Developer Tools:** Standardized way to expose plugin functionality.

Registration Example

```
// First, register a category, or use one of the existing categories.
add_action( 'wp_abilities_api_categories_init', 'wporg_register_category' );
/**
 * Register a custom ability category.
 *
 * @return void
 */
function wporg_register_category() {
	wp_register_ability_category(
		'site-information',
```

```
□□array(
□□□'label'       => __( 'Site Information', 'textdomain' ),
□□□'description' => __( 'Abilities that provide information about the WordPress site.',
'textdomain' ),
□□)
□);
}

// Then, register an ability in that category.
add_action( 'wp_abilities_api_init', 'wporg_register_ability' );
/**
 * Register a custom ability to get site information.
 *
 * @return void
 */
function wporg_register_ability() {
□wp_register_ability(
□□'my-plugin/site-info',
□□array(
□□□'label'               => __( 'Site Info', 'textdomain' ),
□□□'description'         => __( 'Returns information about this WordPress site', 'textdomain' )
□□□'category'            => 'site-information',
□□□'input_schema'        => array(),
□□□'output_schema'       => array(
□□□□'type'       => 'object',
□□□□'properties' => array(
□□□□□'site_name'         => array(
□□□□□□'type'        => 'string',
□□□□□□'description' => __( 'The name of the WordPress site', 'textdomain' ),
□□□□□),
□□□□□'site_url'          => array(
□□□□□□'type'        => 'string',
□□□□□□'description' => __( 'The URL of the WordPress site', 'textdomain' ),
□□□□□),
□□□□□'active_theme'      => array(
□□□□□□'type'        => 'string',
□□□□□□'description' => __( 'The active theme of the WordPress site', 'textdomain' ),
□□□□□),
□□□□□'active_plugins'    => array(
□□□□□□'type'        => 'array',
```

```
□□□□□□□'items'       => array(
□□□□□□□□'type' => 'string',
□□□□□□□),
□□□□□□□'description' => __( 'List of active plugins on the WordPress site', 'textdomain' ),
□□□□□□),
□□□□□□'php_version'       => array(
□□□□□□□'type'        => 'string',
□□□□□□□'description' => __( 'The PHP version of the WordPress site', 'textdomain' ),
□□□□□□),
□□□□□□'wordpress_version' => array(
□□□□□□□'type'        => 'string',
□□□□□□□'description' => __( 'The WordPress version of the site', 'textdomain' ),
□□□□□□),
□□□□),
□□□),
□□□'execute_callback'    => 'wporg_get_siteinfo',
□□□'permission_callback' => function () {
□□□□return current_user_can( 'manage_options' );
□□□},
□□□'meta'                => array(
□□□□'show_in_rest' => true,
□□□),
□□)
□);
}

/**
 * Execute callback to get site information.
 *
 * @return array
 */
function wporg_get_siteinfo() {
□$active_plugins = array();
□foreach ( get_option( 'active_plugins', array() ) as $plugin_path ) {
□□$plugin_data      = get_plugin_data( WP_PLUGIN_DIR . '/' . $plugin_path );
□□$active_plugins[] = $plugin_data['Name'];
□}

□return array(
□□'site_name'         => get_bloginfo( 'name' ),
```

```
□□'site_url'          => get_bloginfo( 'url' ),
□□'active_theme'      => wp_get_theme()->get( 'Name' ),
□□'active_plugins'    => $active_plugins,
□□'php_version'       => PHP_VERSION,
□□'wordpress_version' => get_bloginfo( 'version' ),
□);
}
```

This creates a machine-readable capability that AI systems and automation tools can discover, understand, and execute safely within the bounds of WordPress permissions and validation rules.

□□□ #1
□ □□□ □□□ 2026-08-29 23:36:33 UTC
□ □□□ □□□ 2026-08-29 23:36:33 UTC