Dashboard widgets API

- Overview
  - The main function
  - Usage
  - Action hooks
- Examples
  - Basic usage
  - Forcing your widget to the top
  - Removing default Dashboard Widgets
  - Adding Widgets in the right side
  - Aggregating RSS feeds in the dashboard
- Widget Options
  - Getting Widget Options
  - Get a Single Widget Option
  - Update Widget Options

Added in WordPress Version 2.7, the **Dashboard Widgets API** makes it simple to add new widgets to the administration dashboard.

Doing so requires working knowledge of PHP and the WordPress Plugin API, but to plugin or theme authors familiar with hooking actions and filters, it only takes a few minutes and can be a great way to make your plugin even more useful.



Default Dashboard Widgets

Overview

The main function

The main tool needed to add Dashboard Widgets is the wp_add_dashboard_widget() function. You will find a complete description of this function on that link, but a brief overview is given below.

Usage

```
wp_add_dashboard_widget( $widget_id, $widget_name, $callback, $control_callback,
$callback_args );
```

- `$widget_id`: an identifying slug for your widget. This will be used as its CSS class and its key in the array of widgets.
- `$widget_name`: this is the name your widget will display in its heading.
- `$callback`: The name of a function you will create that will display the actual contents of your widget.
- `$control_callback` (Optional): The name of a function you create that will handle submission of widget options forms, and will also display the form elements.
- `$callback_args` (Optional): Set of arguments for the callback function.

Action hooks

To run the function you will need to hook into the action wp_dashboard_setup via add_action(). For the Network Admin dashboard, use the hook wp_network_dashboard_setup.

```
/**
 * Add a widget to the dashboard.
 *
 * This function is hooked into the 'wp_dashboard_setup' action below.
 */
function wporg_add_dashboard_widgets() {
// Add function here
}
add_action( 'wp_dashboard_setup', 'wporg_add_dashboard_widgets' );
```

Network dashboard:

```
/**
 * Add a widget to the network dashboard.
```

```
 *
 * This function is hooked into the 'wp_network_dashboard_setup' action below.
 */
function wporg_add_network_dashboard_widgets() {
	// Add function here
}
add_action( 'wp_network_dashboard_setup', 'wporg_add_network_dashboard_widgets' );
```

Examples

Basic usage

```
/**
 * Add a widget to the dashboard.
 *
 * This function is hooked into the 'wp_dashboard_setup' action below.
 */
function wporg_add_dashboard_widgets() {
	wp_add_dashboard_widget(
		'wporg_dashboard_widget',                         // Widget slug.
		esc_html__( 'Example Dashboard Widget', 'wporg' ), // Title.
		'wporg_dashboard_widget_render'                   // Display function.
	);
}
add_action( 'wp_dashboard_setup', 'wporg_add_dashboard_widgets' );

/**
 * Create the function to output the content of our Dashboard Widget.
 */
function wporg_dashboard_widget_render() {
	// Display whatever you want to show.
	esc_html_e( "Howdy! I'm a great Dashboard Widget.", "wporg" );
}
```

Forcing your widget to the top

Normally you should just let the users of your plugin put your Dashboard Widget wherever they want by dragging it around. There currently isn’t an easy API way to pre-sort the default widgets, meaning your new widget will always be at the bottom of the list. Until sorting is added to the API its a bit complicated to get around this problem.

Below is an example hooking function that will try to put your widget before the default ones. It does so by manually altering the internal array of metaboxes (of which dashboard widgets are one type) and putting your widget at the top of the list so it shows first.

```
function wporg_add_dashboard_widgets() {
	wp_add_dashboard_widget(
		'wporg_dashboard_widget',
		esc_html__( 'Example Dashboard Widget', 'wporg' ),
		'wporg_dashboard_widget_function'
	);

	// Globalize the metaboxes array, this holds all the widgets for wp-admin.
	global $wp_meta_boxes;

	// Get the regular dashboard widgets array
	// (which already has our new widget but appended at the end).
	$default_dashboard = $wp_meta_boxes['dashboard']['normal']['core'];

	// Backup and delete our new dashboard widget from the end of the array.
	$example_widget_backup = array( 'example_dashboard_widget' =>
$default_dashboard['example_dashboard_widget'] );
	unset( $default_dashboard['example_dashboard_widget'] );

	// Merge the two arrays together so our widget is at the beginning.
	$sorted_dashboard = array_merge( $example_widget_backup, $default_dashboard );

	// Save the sorted array back into the original metaboxes.
	$wp_meta_boxes['dashboard']['normal']['core'] = $sorted_dashboard;
}
add_action( 'wp_dashboard_setup', 'wporg_add_dashboard_widgets' );
```

Unfortunately this only works for people who have never re-ordered their widgets. Once a user has done so their existing preferences will override this and they will have to move your widget to the top for it to stay there.

Removing default Dashboard Widgets

In some situations, especially on multi-user blogs, it may be useful to completely remove widgets from the interface. Each individual user can, by default, turn off any given widget using the *Screen Options* tab at the top, but if you have a lot of non-technical users it might be nicer for them to not see it at all.

To remove dashboard widget, use the remove_meta_box() function. See the example codes below for the required parameters.

These are the names of the default widgets on the dashboard:

```
// Main column (left):
// Browser Update Required
$wp_meta_boxes['dashboard']['normal']['high']['dashboard_browser_nag'];
// PHP Update Required
$wp_meta_boxes['dashboard']['normal']['high']['dashboard_php_nag'];

// At a Glance
$wp_meta_boxes['dashboard']['normal']['core']['dashboard_right_now'];
// Right Now
$wp_meta_boxes['dashboard']['normal']['core']['network_dashboard_right_now'];
// Activity
$wp_meta_boxes['dashboard']['normal']['core']['dashboard_activity'];
// Site Health Status
$wp_meta_boxes['dashboard']['normal']['core']['dashboard_site_health'];

// Side Column (right):
// WordPress Events and News
$wp_meta_boxes['dashboard']['side']['core']['dashboard_primary'];
// Quick Draft, Your Recent Drafts
$wp_meta_boxes['dashboard']['side']['core']['dashboard_quick_press'];
```

Here is an example function that removes the QuickPress widget:

```
// Create the function to use in the action hook
function wporg_remove_dashboard_widget() {
remove_meta_box( 'dashboard_quick_press', 'dashboard', 'side' );
}
// Hook into the 'wp_dashboard_setup' action to register our function
add_action( 'wp_dashboard_setup', 'wporg_remove_dashboard_widget' );
```

The example below removes all Dashboard Widgets:

```
function wporg_remove_all_dashboard_metaboxes() {
	// Remove Welcome panel
	remove_action( 'welcome_panel', 'wp_welcome_panel' );
	// Remove the rest of the dashboard widgets
	remove_meta_box( 'dashboard_primary', 'dashboard', 'side' );
	remove_meta_box( 'dashboard_quick_press', 'dashboard', 'side' );
	remove_meta_box( 'dashboard_site_health', 'dashboard', 'normal' );
	remove_meta_box( 'dashboard_right_now', 'dashboard', 'normal' );
	remove_meta_box( 'dashboard_activity', 'dashboard', 'normal');
}
add_action( 'wp_dashboard_setup', 'wporg_remove_all_dashboard_metaboxes' );
```

Adding Widgets in the right side

The function doesn't allow you to choose where you want your widget to go and will automatically add it to the "core" which is the left side. However you are able to get it on the right side very easily.

You can use the add_meta_box() function instead of `wp_add_dashboard_widget`. Simply specify 'dashboard' in place of the $post_type. For example:

```
add_meta_box(
	'dashboard_widget_id',
	esc_html__( 'Dashboard Widget Title', 'wporg' ),
	'dashboard_widget',
	'dashboard',
	'side', 'high'
);
```

Or, after creating the widget:

```
function wporg_add_dashboard_widget() {
	wp_add_dashboard_widget(
		'wporg_dashboard_widget',
		esc_html__( 'Example Dashboard Widget', 'wporg' ),
		'wporg_dashboard_widget_function'
	);

	// Global the $wp_meta_boxes variable (this will allow us to alter the array).
```

```
	global $wp_meta_boxes;

	// Then we make a backup of your widget.
	$wporg_widget = $wp_meta_boxes['dashboard']['normal']['core']['wporg_dashboard_widget'];

	// We then unset that part of the array.
	unset( $wp_meta_boxes['dashboard']['normal']['core']['wporg_dashboard_widget'] );

	// Now we just add your widget back in.
	$wp_meta_boxes['dashboard']['side']['core']['wporg_dashboard_widget'] = $wporg_widget;
}
add_action( 'wp_dashboard_setup', 'wporg_add_dashboard_widget' );
```

Aggregating RSS feeds in the dashboard

If you need to aggregate RSS in your widget you should take a look at the way the existing plugins are set up with caching in `/wp-admin/includes/dashboard.php`.

Widget Options

WordPress does not provide a built-in way to fetch options for a specific widget. By default, you would need to use `get_option( 'dashboard_widget_options' )` to fetch all widget options and then filter the returned array manually. This section presents some functions that can easily be added to a theme or plugin to help getting and setting of widget options.

Getting Widget Options

This function will fetch all widget options, or only options for a specified widget:

```
/**
 * Gets all widget options, or only options for a specified widget if a widget id is provided.
 *
 * @param string $widget_id Optional. If provided, will only get options for that widget.
 * @return array An associative array
 */
function wporg_get_dashboard_widget_options( $widget_id = '' ) {
    // Fetch ALL dashboard widget options from the db
```

```
    $options = get_option( 'dashboard_widget_options' );

    // If no widget is specified, return everything
    if ( empty( $widget_id ) ) {
        return $options;
    }

    // If we request a widget and it exists, return it
    if ( isset( $options[$widget_id] ) ) {
        return $options[$widget_id];
    }

    // Something went wrong...
    return false;
}
```

Get a Single Widget Option

If you want to easily fetch only a single option (for outputting to a theme), the following function will make that easier.

This example should be used with the previous Getting Widget Options example function.

```
/**
 * Gets one specific option for the specified widget.
 *
 * @param  string $widget_id Widget ID.
 * @param  string $option    Widget option.
 * @param  string $default   Default option.
 *
 * @return string            Returns single widget option.
 */
function wporg_get_dashboard_widget_option( $widget_id, $option, $default = NULL ) {
    $options = wporg_get_dashboard_widget_options( $widget_id );

    // If widget options don't exist, return false.
    if ( ! $options ) {
        return false;
    }
```

```
    // Otherwise fetch the option or use default
    if ( isset( $options[$option] ) && ! empty( $options[$option] ) ) {
        return $options[$option];
    } else {
        return ( isset( $default ) ) ? $default : false;
    }
}
```

Update Widget Options

This function can be used to easily update all of a widget's options. It can also be used to add a widget option non-destructively. Simply set the $add_option argument to true, and this will **NOT overwrite** any existing options (although it will add any missing ones).

```
/**
 * Saves an array of options for a single dashboard widget to the database.
 * Can also be used to define default values for a widget.
 *
 * @param string $widget_id  The name of the widget being updated
 * @param array $args        An associative array of options being saved.
 * @param bool $add_only     Set to true if you don't want to override any existing options.
 */
function update_dashboard_widget_options( $widget_id , $args = array(), $add_only = false ) {
    // Fetch ALL dashboard widget options from the db...
    $options = get_option( 'dashboard_widget_options' );

    // Get just our widget's options, or set empty array.
    $widget_options = ( isset( $options[$widget_id] ) ) ? $options[$widget_id] : array();

    if ( $add_only ) {
        // Flesh out any missing options (existing ones overwrite new ones).
        $options[$widget_id] = array_merge( $args, $widget_options );
    } else {
        // Merge new options with existing ones, and add it back to the widgets array.
        $options[$widget_id] = array_merge( $widget_options, $args );
    }

    // Save the entire widgets array back to the db.
    return update_option( 'dashboard_widget_options', $options );
```

```
}
```

□□□ #2
□ □□□ □□□ 2026-08-29 23:36:33 UTC
□ □□□ □□□ 2026-08-29 23:36:33 UTC