



The Handbook

Codex

[http://codex.wordpress.org/Adding Administration Menus](http://codex.wordpress.org/Adding_Administration_Menus)

Version Date

6 August 2005

Adding Administration Menus

Introduction

As a [Plugin author](#), you may have the need to provide a way for users to configure options for your plugin. Rather than forcing users - who often have sparse programming knowledge - to sift through your plugin code, WordPress provides some easy methods to embed user interface elements into the admin console.

NOTE: This functionality is being revamped for WordPress 1.5.1. Currently, the documentation here describes the improved method which will be available in that release. Everything should work, though, in version 1.5 except for some of the top level menu addition functionality and subsequent submenu additions to that new top level menu.

Admin Interface Plugin Guidelines

When creating an interface for your plugin, conceptualize the intention of the interface that you plan to provide. Use this guide to determine where in the submenus your new menu item should be included.

[Options](#)

Displays plugin options that only administrators should view. The "how" of data management.

[Manage](#)

Displays management controls for finite tasks like links, posts, images, etc. The "what" of data management.

[Dashboard](#)

Displays general overview/summary/status information, but offers no options or ways to alter data.

[Plugins](#)

Displays controls dealing with plugin management, not configuration options for a plugin itself.

[Presentation](#)

Displays controls for manipulation of theme/style files.

[Write](#)

Displays tools for writing content onto the site. Might contain a plugin to modify sidebar configuration.

[Users](#)

Displays controls for user management. A plugin that lets you directly select a user level via a dropdown might go here.

Keep these guidelines in mind for other menus in the WordPress admin, and make sure your submenu is appropriate to their content.

Regarding Top Level Menus

It is also possible, but *strongly discouraged*, to create top-level menus in the admin console. New top-level menus should only be used for adding brand new features to WordPress; features that are not innate. Gallery management, database administration -- Stuff that would be packed too tightly into a single submenu. Just because you're not a GUI wizard doesn't mean you can't ask how to fit your features comfortably into a single well-crafted submenu page.

If you would need two submenus under a single top-level menu (which is not possible with the current code), you might be a candidate for a new top-level menu, but think carefully about this before you commit to it. WordPress users like that their admin consoles are clean of clutter.

Admin Menu Functions

Several WordPress functions help plugin developers create new menus and pages within the admin console. These functions are primarily located within the file `wp-admin/admin-functions.php`, but the application of menus is handled by `admin.php`, `menu-header.php`, and `menu.php`.

These functions should be called from inside an 'admin_menu' plugin hook sink function. Do not call these functions from the global namespace because the menu structure may not yet have been created by WordPress.

add_submenu_page()

```
add_submenu_page(parent, page_title, menu_title, access_level, file, [function]);
```

Use `add_submenu_page()` to add a submenu to a top-level menu in the admin console. This function should be used to add submenu pages to existing core top-level menus and to custom top-level menus.

`add_submenu_page()` is the basic function for adding new submenus. Other submenu functions are wrappers for this function, and have all but one of the same parameters.

Parameter values:

`parent`

The filename of the core WordPress admin file that supplies the top-level menu in which you want to insert your submenu. (One of these: `index.php`, `post.php`, `edit.php`, `themes.php`, `plugins.php`, etc.)

`page_title`

The title value that will be used for the page when the submenu is active.

`menu_title`

The text of the menu option in the submenu.

`access_level`

The minimum [user level](#) required to display and activate this menu page.

`file`

The file that handles the display of the menu page content or a unique identifier for a submenu of a custom top-level menu.

`function`

Optional. The function that displays the page content for the menu page.

If the optional *function* parameter is not supplied, then WordPress will include the file referenced by the *file* parameter into the content area of the page. Without code to detect whether your plugin functions have been declared, this will likely cause PHP errors when all of your plugin functions are redeclared. You can use the `is_plugin_page()` function to mitigate this issue; however, using the *function* parameter is recommended.

add_options_page()

```
add_options_page(page_title, menu_title, access_level, file, [function]);
```

Use `add_options_page()` to add a submenu to the Options top-level menu. This function is a wrapper for `add_submenu_page()`. See `add_submenu_page()` above for a parameter reference.

add_management_page()

```
add_management_page(page_title, menu_title, access_level, file, [function]);
```

Use `add_management_page()` to add a submenu to the Manage top-level menu. This function is a wrapper for `add_submenu_page()`. See `add_submenu_page()` above for a parameter reference.

add_menu_page()

```
add_menu_page(page_title, menu_title, access_level, file, [function]);
```

Use `add_menu_page()` to add a top-level menu. **Using this function is not recommended for most plugins!** Choose a main menu item and use `add_submenu_page()` or one of its derivatives instead, if at all possible.

Parameter values:

`page_title`

The title value that will be used for the page when the submenu is active.

`menu_title`

The text of the menu option to use as the top-level menu.

`access_level`

The minimum [user level](#) required to display and activate this menu page.

`file`

The file that handles the display of the menu page content.

`function`

Optional. The function that displays the page content for the menu page.

If the optional *function* parameter is not supplied, then WordPress will include the file referenced by the *file* parameter into the content area of the page. Without code to detect whether your plugin functions have been declared, this will likely cause PHP errors when all of your plugin functions are redeclared. You can use the `is_plugin_page()` function to mitigate this issue; however, using the *function* parameter is recommended.

Example

Here is an example of a WordPress plugin that inserts new menus into various places:

```
<?php
/*
Plugin Name: Menu Test
Plugin URI: http://wordpress.org
Description: Menu Test
Author: Nobody
Author URI: http://example.com
*/

// mt_add_pages() is the sink function for the 'admin_menu' hook
function mt_add_pages() {
    // Add a new menu under Options:
    add_options_page('Test Options', 'Test Options', 8, __FILE__,
'mt_options_page');

    // Add a new menu under Manage:
    add_management_page('Test Manage', 'Test Manage', 8, __FILE__,
'mt_manage_page');

    // Add a submenu to the Dashboard:
    add_submenu_page('index.php', 'Glove Compartment', 'Glove Compartment', 8,
'glove-compartment', 'mt_glove_page');

    // Add a new top-level menu (ill-advised):
    add_menu_page('Test Toplevel', 'Test Toplevel', 8, __FILE__,
'mt_toplevel_page');

    // Add a submenu to the custom top-level menu:
    add_submenu_page(__FILE__, 'Test Sublevel', 'Test Sublevel', 8, 'sub-page',
'mt_sublevel_page');

    // Add a second submenu to the custom top-level menu:
    add_submenu_page(__FILE__, 'Test Sublevel 2', 'Test Sublevel 2', 8, 'sub-
page2', 'mt_sublevel_page2');
}

// mt_options_page() displays the page content for the Test Options submenu
function mt_options_page() {
    echo "<h2>Test Options</h2>";
}

// mt_manage_page() displays the page content for the Test Manage submenu
function mt_manage_page() {
    echo "<h2>Test Manage</h2>";
}

// mt_toplevel_page() displays the page content for the custom Test Toplevel
menu
function mt_toplevel_page() {
    echo "<h2>Test Toplevel</h2>";
}

// mt_sublevel_page() displays the page content for the first submenu
// of the custom Test Toplevel menu
function mt_sublevel_page() {
    echo "<h2>Test Sublevel</h2>";
}
```

```
// mt_sublevel_page2() displays the page content for the second submenu
// of the custom Test Toplevel menu
function mt_sublevel_page2() {
    echo "<h2>Test Sublevel 2</h2>";
}

// mt_glove_page() displays the page content for the Glove Compartment submenu
function mt_glove_page() {
    echo "<h2>Test Glove Compartment</h2>";
}

// Insert the mt_add_pages() sink into the plugin hook list for 'admin_menu'
add_action('admin_menu', 'mt_add_pages');
?>
```