Action Reference

- Actions Run During a Typical Request
- Actions Run During an Admin Page Request
- Post, Page, Attachment, and Category Actions (Admin)
  - Taxonomy and Terms
- Comment, Ping, and Trackback Actions
- Blogroll Actions
- Feed Actions
- Template Actions
- Administrative Actions
- Advanced Actions
- Admin Login Actions
- Further Reading

This is a (hopefully) comprehensive list of action hooks available in WordPress version 2.1 and above. For more information:

- To learn more about what filter and action hooks are, see Plugin API.
- To learn about writing plugins in general, see Plugin Handbook.
- For a reference list of filter hooks, see Plugin API/Filter Reference.

(If you want to add to or clarify this documentation, please follow the style of the existing entries. Describe when the action runs, and if the action function takes arguments, describe the arguments.)

Actions Run During a Typical Request

These actions are called when a logged-in user opens the home page, this list may be outdated. This list may show only the first time each action is called, and in many cases no function is hooked to the action. Themes and plugins can cause actions to be called multiple times and at differing times during a request. As proof of this, you can see action calls specific to the Twenty Eleven theme on this list. Cron tasks may also fire when a user visits the site, adding additional action calls. This list should be viewed as a guideline or approximation of WordPress action execution order, and not a concrete specification.

Actions are called with the function `do_action()`, except those marked (ref array), which are called with the function `do_action_ref_array()`.

muplugins_loaded	After must-use plugins are loaded.
registered_taxonomy	For category, post_tag, *etc.*
registered_post_type	For post, page, *etc.*
plugins_loaded	After active plugins and before pluggable functions are loaded.
sanitize_comment_cookies	When comment cookies are sanitized.
setup_theme	Before the theme is loaded.
load_textdomain	For the `default` domain
after_setup_theme	Generally used to initialize theme settings/options. This is the **first action hook available to themes** , triggered immediately after the active theme's *functions.php* file is loaded. `add_theme_support()` should be called here, since the `init` action hook is too late to add some features. At this stage, the current user is not yet authenticated.
auth_cookie_malformed	
auth_cookie_valid	
set_current_user	
init	Typically used by plugins to initialize. The current user is already authenticated by this time.
└─ widgets_init	Used to register sidebars. Fired at 'init' priority 1 (and so before 'init' actions with priority ≥ 1!)
register_sidebar	For each sidebar and footer area
wp_register_sidebar_widget	For each widget
wp_default_scripts	(ref array)

wp_default_styles	(ref array)
admin_bar_init	
add_admin_bar_menus	
wp_loaded	After WordPress is fully loaded
parse_request	Allows manipulation of HTTP request handling (ref array)
send_headers	Allows customization of HTTP headers (ref array)
parse_query	After query variables are set (ref array)
pre_get_posts	Exposes the query-variables object before a query is executed. (ref array)
posts_selection	Used by caching plugins.
wp	After WP object is set up (ref array)
template_redirect	Before determining which template to load.
get_header	Before the header template file is loaded. Not relevant for block themes.
wp_enqueue_scripts	When scripts and styles are enqueued.
twentyeleven_enqueue_color_scheme	(Specific to Twenty Eleven)
wp_head	Used to print scripts or data in the head tag on the front end.
wp_print_styles	Before styles in the $handles queue are printed.
wp_print_scripts	Before scripts in the $handles queue are printed.
get_search_form	
loop_start	(ref array)
the_post	(ref array) Allows modification of the post object immediately after query
get_template_part_content	Template part for the content
loop_end	(ref array)
get_sidebar	Before the sidebar template file is loaded.
dynamic_sidebar	Before a widget’s display callback is called.
get_search_form	

pre_get_comments	(ref array)
wp_meta	Before displaying echoed content in the sidebar.
get_footer	Before the footer template file is loaded. Not relevant for block themes.
get_sidebar	Before the sidebar template file is loaded. Not relevant for block themes.
twentyeleven_credits	(Specific to Twenty Eleven)
wp_footer	Before determining which template to load.
wp_print_footer_scripts	When footer scripts are printed.
admin_bar_menu	(ref array)
wp_before_admin_bar_render	Before the admin bar is rendered.
wp_after_admin_bar_render	After the admin bar is rendered.
shutdown	Before PHP execution is about to end.

Actions Run During an Admin Page Request

These actions are run when a logged-in user opens the Posts page. This list shows only the first time an action is called, and in many cases no function is hooked to the action. Each admin page has a different list of actions depending upon the purpose of the page and the plugins installed. This list should be viewed as a guide line or approximation, and not a concrete specification.

In these actions, `(hookname)` depends on the page. For the Posts page it is *edit.php*, or for a theme's Background page it is `appearance_page_custom-background`.

Actions are called with the function `do_action()`, except those marked (ref array), which are called with the function `do_action_ref_array()`.

muplugins_loaded	After must-use plugins are loaded
registered_taxonomy	For category, post_tag, *etc.*
registered_post_type	For post, page, *etc.*

plugins_loaded	After active plugins and pluggable functions are loaded
sanitize_comment_cookies	
setup_theme	
load_textdomain	For domain `default`
after_setup_theme	At this stage, the current user is not yet authenticated.
load_textdomain	For domain `twentyeleven`
auth_cookie_valid	
set_current_user	
init	Typically used by plugins to initialize. The current user is already authenticated by this time.
└─ widgets_init	Used to register sidebars. This is fired at 'init', with a priority of 1.
register_sidebar	For each sidebar
wp_register_sidebar_widget	For each widget
wp_default_scripts	(ref array)
wp_default_styles	(ref array)
admin_bar_init	
add_admin_bar_menus	
wp_loaded	After WordPress is fully loaded
auth_cookie_valid	
auth_redirect	
_admin_menu	See also: _user_admin_menu, _network_admin_menu
admin_menu	See also: user_admin_menu, network_admin_menu
admin_init	
current_screen	
load-{$page_hook}	
send_headers	Where custom HTTP headers can be added

pre_get_posts	Exposes the query-variables object before a query is executed. (ref array)
posts_selection	
wp	After WP object is set up (ref array)
admin_xml_ns	
admin_xml_ns	
admin_enqueue_scripts	
admin_print_styles-{$hook_suffix}	
admin_print_styles	
admin_print_scripts-{$hook_suffix}	
admin_print_scripts	
wp_print_scripts	
admin_head-{$hook_suffix}	
admin_head	
admin_menu	
in_admin_header	
admin_notices	
all_admin_notices	
restrict_manage_posts	
the_post	(ref array)
pre_user_query	(ref array)
in_admin_footer	
admin_footer	
admin_bar_menu	(ref array)
wp_before_admin_bar_render	
wp_after_admin_bar_render	
admin_print_footer_scripts	

admin_footer-{$hook_suffix}	Admin page footer
shutdown	PHP execution is about to end
wp_dashboard_setup	Allows customization of admin Dashboard

Post, Page, Attachment, and Category Actions (Admin)

post_submitbox_misc_actions : Runs when an editing page gets generated to add some content (eg. fields) to the submit box (where the publish button is shown). No function arguments.

add_attachment : Runs when an attached file is first added to the database. Action function arguments: attachment ID.

add_category : Same as `create_category`.

{$taxonomy}_add_form_fields : Runs when a taxonomy add form is cerated in admin. Useful to add a field in this form before the submit button. For example `category_add_form_fields`.

{$taxonomy}_edit_form : Runs when taxonomy term edit form is created in admin. Useful to add a new field to this form.

clean_post_cache : Runs when post cache is cleaned. Action function arguments: post ID. See `clean_post_cache()`.

create_{$taxonomy} : Runs when a new taxonomy term is created. Action function arguments: term ID.

delete_attachment : Runs just before an attached file is deleted from the database. Action function arguments: attachment ID. ''(Prior to version 2.8 this hook was triggered after attachment was deleted.)

delete_{$taxonomy} : Runs just after a taxonomy term is deleted from the database and its corresponding links/posts are updated to remove the term. Action function arguments: Term ID.

wp_trash_post : Runs when a post or page is about to be trashed. Action function arguments: post or page ID.

trashed_post : Runs just after a post or page is trashed. Action function arguments: post or page ID.

untrash_post : Runs just before undeletion, when a post or page is restored. Action function arguments: post or page ID.

untrashed_post : Runs just after undeletion, when a post or page is restored. Action function arguments: post or page ID.

before_delete_post : Runs when a post or page is about to be deleted. Comments, attachments and metadata are still available. Action function arguments: post or page ID.

delete_post : Runs when a post or page is about to be deleted. Comments, attachments and metadata are already deleted. Action function arguments: post or page ID.

deleted_post : Runs just after a post or page is deleted. Action function arguments: post or page ID.

edit_attachment : Runs when an attached file is edited/updated to the database. Action function arguments: attachment ID.

edit_category : Runs when a category is updated/edited, including when a post or blogroll link is added/deleted or its categories are updated (which causes the count for the category to update). Action function arguments: category ID.

edit_post : Runs when a post or page is updated/edited, including when a comment is added or updated (which causes the comment count for the post to update). Action function arguments: post or page ID.

pre_post_update : Runs just before a post or page is updated. Action function arguments: post or page ID.

post_updated : Runs after a post or page is updated. Action function arguments: post or page ID, WP_Post object of the post before the update and after the update.

transition_post_status : Runs when any post status transition occurs. Action function arguments: `$new_status`, `$old_status`, `$post` object. (See also Post Status Transitions.)

(old status)*to*(new status) : Runs when a post changes status from `$old_status` to `$new_status`. Action function arguments: `$post` object. (See also Post Status Transitions.)

{$new_status}_{$post->post_type} : Runs when a post of type `$post_type` is transitioned to `$status` from any other status. Action function arguments: post ID, `$post object`. (See also Post

Status Transitions.)

publish_post (not deprecated) : Runs when a post is published, or if it is edited and its status is changed to “published”. This action hook conforms to the {$new_status}_{$post->post_type} action hook type. Action function arguments: post ID, `$post object`. (See also Post Status Transitions.)

publish_page : Runs when a page is published, or if it is edited and its status is changed to “published”. This action hook conforms to the {$new_status}_{$post->post_type} action hook type. Action function arguments: post ID, `$post object`. (See also Post Status Transitions.)

publish_phone : Runs just after a post is added via email. Action function argument: post ID.

save_post : Runs whenever a post or page is created or updated, which could be from an import, post/page edit form, xmlrpc, or post by email. Action function arguments: post ID and post object. Runs after the data is saved to the database. Note that post ID may reference a post revision and not the last saved post. Use `wp_is_post_revision()` to get the ID of the real post.

updated_postmeta : Runs when a metadata has been updated.

wp_insert_post : Same as `save_post`, runs immediately afterwards.

xmlrpc_publish_post : Runs when a post is published via XMLRPC request, or if it is edited via XMLRPC and its status is “published”. Action function arguments: post ID.

Taxonomy and Terms

create_term : Runs after a new term is created, before the term cache is cleaned.

created_term : Runs after a new term is created, and after the term cache has been cleaned.

create_$taxonomy : Runs after a new term is created for a specific taxonomy.

created_$taxonomy : Runs after a new term in a specific taxonomy is created, and after the term cache has been cleaned.

add_term_relationship (Since version 2.9.0) : Runs before an object-term relationship is added.

added_term_relationship (Since version 2.9.0) : Runs after an object-term relationship is added.

set_object_terms (Since version 2.8.0) : Runs after an object's terms have been set.

edit_terms (Since version 2.9.0) : Runs before the given terms are edited.

edited_terms : Runs after saving taxonomy/category change in the database.

edit_term_taxonomy : Runs before a term-taxonomy relationship is updated.

edited_term_taxonomy : Runs after a term-taxonomy relationship is updated.

edit_term_taxonomies (Since version 2.9.0) : Runs before a term to delete's children are reassigned a parent.

edited_term_taxonomies (Since version 2.9.0) : Runs after a term to delete's children are reassigned a parent.

edit_$taxonomy : Runs after a term is edited for a specific taxonomy.

edited_$taxonomy : Runs after a term in a specific taxonomy is edited, and after the term cache has been cleaned.

pre_delete_term (Since version 4.1.0) : Runs before any modifications are made to posts or terms.

delete_term_taxonomy (Since version 2.9.0) : Runs before a term taxonomy ID is deleted from database (after having change chidren's term).

deleted_term_taxonomy (Since version 2.9.0) : Runs after a term taxonomy ID is deleted.

delete_term (Since version 2.5.0) : Runs after a term is deleted from the database and the cache is cleaned. (Paramètres : $Term_ID, $Term_taxonomy_ID, $Taxonomy_slug, $already_deleted_term)

delete_$taxonomy (Since version 2.3.0) : Runs after a term in a specific taxonomy is deleted. (Paramètres : $Term_ID, $Term_taxonomy_ID, $already_deleted_term)

delete_term_relationships (Since version 2.9.0) : Runs before an object-term relationship is deleted.

deleted_term_relationships (Since version 2.9.0) : Runs after an object-term relationship is deleted.

clean_object_term_cache (Since version 2.5.0) : Runs after the object term cache has been cleaned.

clean_term_cache (Since version 2.5.0) : Runs after each taxonomy's term cache has been cleaned.

split_shared_term (Since version 4.2.0) : Runs after a previously shared taxonomy term is split into two separate terms.

pre_term_{$field} : Runs before a taxonomy term's data is saved to the database. For example, `pre_term_description`.

pre_{$taxonomy}_{$field} : Runs before a term's field is saved to the database. For example, `pre_category_description`.

Comment, Ping, and Trackback Actions

comment_closed : Runs when the post is marked as not allowing comments while trying to display comment entry form. Action function argument: post ID.

comment_id_not_found : Runs when the post ID is not found while trying to display comments or comment entry form. Action function argument: post ID.

comment_flood_trigger : Runs when a comment flood is detected, just before `wp_die` is called to stop the comment from being accepted. Action function arguments: time of previous comment, time of current comment.

comment_{$old_status}_*to*_{$new_status} : Runs when a comment status transition occurs. Action function arguments: Comment object.

comment_on_draft : Runs when the post is a draft while trying to display a comment entry form or comments. Action function argument: post ID.

comment_post : Runs just after a comment is saved in the database. Action function arguments: comment ID, approval status ("spam", or 0/1 for disapproved/approved).

edit_comment : Runs after a comment is updated/edited in the database. Action function arguments: comment ID.

delete_comment : Fires immediately before a comment is deleted from the database. Action function arguments: comment ID.

deleted_comment : Fires immediately after a comment is deleted from the database. Action function arguments: comment ID.

trash_comment : Fires immediately before a comment is sent to the Trash. Action function arguments: comment ID.

trashed_comment : Fires immediately after a comment is sent to Trash. Action function arguments: comment ID.

untrash_comment : Fires immediately before a comment is restored from the Trash. Action function arguments: comment ID.

untrashed_comment : Fires immediately after a comment is restored from the Trash. Action function arguments: comment ID.

spam_comment : Fires immediately before a comment is marked as Spam. Action function arguments: comment ID.

spammed_comment : Fires immediately after a comment is marked as Spam. Action function arguments: comment ID.

unspam_comment : Fires immediately before a comment is unmarked as Spam. Action function arguments: comment ID.

unspammed_comment : Fires immediately after a comment is unmarked as Spam. Action function arguments: comment ID.

pingback_post : Runs when a ping is added to a post. Action function argument: comment ID.

pre_ping : Runs before a ping is fully processed. Action function arguments: array of the post links to be processed, and the “pung” setting for the post.

trackback_post : Runs when a trackback is added to a post. Action function argument: comment ID.

wp_blacklist_check : Runs to check whether a comment should be blacklisted. Action function arguments: author name, author email, author URL, comment text, author IP address, author’s user agent (browser). Your function can execute a `wp_die` to reject the comment, or perhaps modify one of the input arguments so that it will contain one of the blacklist keywords set in the WordPress options.

wp_insert_comment : Runs whenever a comment is created.

wp_set_comment_status : Runs when the status of a comment changes. Action function arguments: comment ID, status string indicating the new status (“delete”, “approve”, “spam”, “hold”).

Blogroll Actions

add_link : Runs when a new blogroll link is first added to the database. Action function arguments: link ID.

delete_link : Runs when a blogroll link is deleted. Action function arguments: link ID.

edit_link : Runs when a blogroll link is edited. Action function arguments: link ID.

Feed Actions

atom_entry : Runs just after the entry information has been printed (but before closing the entry tag) for each blog entry in an atom feed.

atom_head : Runs just after the blog information has been printed in an atom feed, just before the first entry.

atom_ns : Runs inside the root XML element for an atom feed (to add namespaces).

commentrss2_item : Runs just after a single comment’s information has been printed in a comment feed (but before closing the item tag). Action function arguments: comment ID, post ID.

do_feed_{$feed} : Runs when a feed is generated, where `feed` is the type of feed (`rss2, atom, rdf` , etc.). Use `priority` less than 10 to run **before** printing the feed. Action function argument: `true` (the feed is for comments) or `false` (it is for posts).

rdf_header : Runs just after the blog information has been printed in an RDF feed, just before the first entry.

rdf_item : Runs just after the entry information has been printed (but before closing the item tag) for each blog entry in an RDF feed.

rdf_ns : Runs inside the root XML element in an RDF feed (to add namespaces).

rss_head : Runs just after the blog information has been printed in an RSS feed, just before the first entry.

rss_item : Runs just after the entry information has been printed (but before closing the item tag) for each blog entry in an RSS feed.

rss2_head : Runs just after the blog information has been printed in an RSS 2 feed, just before the first entry.

rss2_item : Runs just after the entry information has been printed (but before closing the item tag) for each blog entry in an RSS 2 feed.

rss2_ns : Runs inside the root XML element in an RSS 2 feed (to add namespaces).

Template Actions

after_setup_theme : Runs during a themes initialization. Is generally used to perform basic setup, registration, and init actions for a theme.

comment_form : Runs at the bottom of a comment form rendered by comment_form(), right before the closing

comment_form_after : Runs after the comment form is rendered by comment_form(), right after the closing

do_robots : Runs when the template file chooser determines that it is a robots.txt request.

do_robotstxt : Runs in the `do_robots()` function before it prints out the Disallow lists for the robots.txt file.

get_footer : Runs when the template calls the `get_footer()` function, just before the `footer.php` template file is loaded.

get_header : Runs when the template calls the `get_header()` function, just before the `header.php` template file is loaded.

switch_theme : Runs when the blog's theme is changed. Action function argument: name of the new theme. If used in a theme, it only works if the theme that adds action is the one being disabled.

after_switch_theme : Runs when the blog's theme is changed. Action function argument: name of the new theme. If used in a theme, it only works if the theme that adds action is the one being enabled. Can be used to run certain code when enabling a theme.

load-themes.php : Runs when the theme is activate or deactivate (replace by an other).

template_redirect : Runs before the determination of the template file to be used to display the requested page.

wp_footer : Runs when the template calls the `wp_footer()` function, generally near the bottom of the blog page.

wp_head : Runs when the template calls the `wp_head()` function. This hook is generally placed near the top of a page template between

wp_meta : Runs when the *sidebar.php* template file calls the `wp_meta()` function, to allow the plugin to insert content into the sidebar.

wp_print_scripts : Runs just before WordPress prints registered JavaScript scripts into the page header.

Administrative Actions

activate_{$plugin} : Runs when the plugin is first activated. See

Function_Reference/register_activation_hook.

activity_box_end : Runs at the end of the activity box on the admin Dashboard screen.

add_category_form_pre : Runs before the add category form is put on the screen in the admin menus.

add_option_{$option} : Runs after a WordPress option has been added by the add_option() function. Action function arguments: option name, option value. You must add an action for the specific options that you want to respond to, such as 'add_option_foo' to respond when option "foo" has been added.

add_option : Runs before an option gets added to the database.

added_option : Runs after an an option has been added.

admin_head :Runs in the HTML

admin_head-{$hook_suffix} : Runs in the HTML

admin_init : Runs at the beginning of every admin page before the page is rendered. See *wp-admin/admin.php*, *wp-admin/admin-post.php*, and *wp-admin/admin-ajax.php*.

admin_footer-{$hook_suffix} : Runs at the end of the

admin_post_{$action} : also: admin_post_nopriv_{$action} – Runs a handler for an unspecified GET or POST request.

admin_footer : Runs at the end of the admin panel inside the body tag

admin_enqueue_scripts : Runs in the HTML header so a plugin or theme can enqueue JavaScript and CSS to all admin pages.

admin_print_scripts : Runs in the HTML header so a plugin can add JavaScript scripts to all admin pages.

admin_print_scripts-{$hook_suffix} : Runs to print JavaScript scripts in the HTML head section of a specific plugin-generated admin page. The (page_hook) is returned when using any of the functions that add plugin menu items to the admin menu: add_management_page() , add_options_page() , etc. Example:

```
function myplugin_menu() {
   if ( function_exists( 'add_management_page' ) ) {
      $page = add_management_page( 'myplugin', 'myplugin', 'manage_options', 'myplugin_slug', '
myplugin_admin_page' );
      add_action( "admin_print_scripts-$page", 'myplugin_admin_head' );
   }
}
```

admin_print_styles : Runs in the HTML header so a plugin can add CSS/Stylesheets to all admin pages.

admin_print_styles-{$hook_suffix} : Runs when styles should be enqueued with `wp_enqueue_style()` for a particular admin page. Use the return value of a function such as `add_submenu_page()` to determine the value of *(page_hook)*.

check_passwords : Runs to validate the double-entry of password when creating a new user. Action function arguments: array of login name, first password, second password.

dbx_post_advanced : Runs at the bottom of the “advanced” section on the page editing screen in the admin menus.

dbx_post_sidebar : Runs at the bottom of the sidebar on the page editing screen in the admin menus.

dbx_post_advanced : Runs at the bottom of the “advanced” section on the post editing screen in the admin menus.

dbx_post_sidebar : Runs at the bottom of the sidebar on the post editing screen in the admin menus. Use `add_meta_box()` in WordPress 2.5 and higher.

deactivate_{$plugin} : Runs when a plugin is deactivated.

delete_option_{$option} : Runs after a WordPress option has been deleted by the delete_option() function. Action function arguments: option name. You must add an action for the specific options that you want to respond to, such as ‘delete_option_foo’ to respond when option “foo” has been deleted.

delete_option : Runs before an option gets deleted from the database.

deleted_option : Runs after an an option has been deleted.

delete_user : Runs when a user is deleted. Action function arguments: user ID.

edit_category_form : Runs after the add/edit category form is put on the screen (but before the end of the HTML form tag).

edit_category_form_pre : Runs before the edit category form is put on the screen in the admin menus.

edit_tag_form : Runs after the add/edit tag form is put on the screen (but before the end of the HTML form tag).

edit_tag_form_pre : Runs before the edit tag form is put on the screen in the admin menus.

edit_form_top : Runs inside the form before the title on WordPress post edit screen (and Custom Post Types), but after the inital hidden fields (user_ID, action, etc.).

edit_form_after_title : Runs after the title on WordPress post edit screen (and Custom Post Types) but before the built in WordPress content area.

edit_form_after_editor : Runs just after the WordPress post editor but before all other meta boxes. also available in Custom Post Types.

edit_form_advanced : Runs just before the “advanced” section of the post editing form in the admin menus.

edit_page_form : Runs just before the “advanced” section of the page editing form in the admin menus.

edit_user_profile : Runs near the end of the user profile editing screen in the admin menus.

load-{$pagenow} : Runs when an administration menu page is loaded. This action is not usually added directly — see Adding Administration Menus for more details of how to add admin menus. If you do want to use it directly, the return value from `add_options_page()` and similar functions gives you the “(page)” part of the action name.

login_form : Runs just before the end of the login form.

login_head : Runs just before the end of the HTML head section of the login page.

lost_password : Runs before the “retrieve your password by email” form is printed on the login screen.

lostpassword_form : Runs at the end of the form used to retrieve a user’s password by email, to allow a plugin to supply extra fields.

lostpassword_post : runs when the user has requested an email message to retrieve their password, to allow a plugin to modify the PHP `$_POST` variable before processing.

manage_link_custom_column : Runs when there is an unknown column name for an admin screen. Action function arguments: column name, link ID. See also filter id}_columns in the Plugin API/Filter Reference, which adds custom columns.

manage_posts_custom_column : Runs when there is an unknown column name for the managing posts admin screen. Action function arguments: column name, post ID. See also filter manage_posts_columns in the Plugin API/Filter Reference, which adds custom columns. (see Scompt’s tutorial for examples and use.)

manage_pages_custom_column : Runs when there is an unknown column name for the managing pages admin screen. Action function arguments: column name, page ID. See also filter manage_pages_columns in the Plugin API/Filter Reference, which adds custom columns.

manage_media_custom_column : Runs when there is an unknown column name for the managing media admin screen. Action function arguments: column name, page ID. See also filter manage_media_columns in the Plugin API/Filter Reference, which adds custom columns.

manage_{$post->post_type}_posts_custom_column : Runs when there is an unknown column name for the managing custom post type admin screen. Action function arguments: column name, post ID. See also filter manage_{$post_type}_posts_columns, which adds custom columns for custom post types.

password_reset : Runs before the user's password is reset to a random new password.

personal_options_update : Runs when a user updates personal options from the admin screen.

plugins_loaded : Runs after all plugins have been loaded.

profile_personal_options : Runs at the end of the Personal Options section of the user profile editing screen.

profile_update : Runs when a user's profile is updated. Action function argument: user ID.

quick_edit_custom_box : Runs when there is an unknown column name when creating the quick editor.

register_form : Runs just before the end of the new user registration form.

register_post : Runs before a new user registration request is processed.

restrict_manage_posts : Runs before the list of posts to edit is put on the screen in the admin menus.

retrieve_password : Runs when a user's password is retrieved, to send them a reminder email. Action function argument: login name.

set_current_user : Runs after the user has been changed by the default `wp_set_current_user()` function. Note that `wp_set_current_user()` is also a "pluggable" function, meaning that plugins can override it; see Plugin API).

show_user_profile : Runs near the end of the user profile editing screen.

sidebar_admin_page : Runs after the main content on the widgets admin page.

sidebar_admin_setup : Runs early when editing the widgets displayed in sidebars.

update_option_{$option} : Runs after a WordPress option has been updated by the `update_option()` function. Action function arguments: old option value, new option value. You must add an action for the specific options that you want to respond to, such as 'update_option_foo' to respond when option "foo" has been updated.

update_option : Runs before an option gets updated to the database.

updated_option : Runs after an option has been updated.

user_new_form : Runs near the end of the "Add New" user screen. Action function argument: Passes the string "add-existing-user" on multisite or "add-new-user" on single site and for network admins.

user_profile_update_errors : Runs just before updated user details are committed to the database.

wpmu_new_user : Runs when a user's profile is first created in a Multisite environment. Action function argument: user ID. If not in Multisite then use user_register.

user_register : Runs when a user's profile is first created. Action function argument: user ID.

welcome_panel : Allows you to hide the Welcome Panel in the Dashboard. This is also a smart filter, which hides the related Screen Option.

wp_ajax_{$action} : also: wp_ajax_nopriv_{$action} – Runs to do an unknown type of AJAX request handler.

wp_authenticate : Runs to authenticate a user when they log in. Action function arguments: array with user name and password.

wp_login : Runs when a user logs in.

wp_logout : Runs when a user logs out.

Advanced Actions

This section contains actions related to the queries WordPress uses to figure out what posts to display, the WordPress loop, activating plugins, and other fundamental-level WordPress code.

activated_plugin : Runs any time any plugin is successfully activated

add_meta_boxes : Runs when “edit post” page loads. (**3.0+**)

admin_menu : Runs after the basic admin panel menu structure is in place.

network_admin_notices : Runs after the admin menu is printed to network admin screens.

user_admin_notices : Runs after the admin menu is printed to user admin screens.

admin_notices : Runs after the admin menu is printed to screens that aren’t network- or user-admin screens.

all_admin_notices : Runs after the admin menu is printed to all screens.

blog_privacy_selector : Runs after the default blog privacy options are printed on the screen.

check_admin_referer : Runs in the default `check_admin_referrer()` function after the nonce has been checked for security purposes, to allow a plugin to force WordPress to die for extra security reasons. Note that `check_admin_referrer` is also a “pluggable” function, meaning that plugins can override it; see Plugin API).

check_ajax_referer : Runs in the default `check_ajax_referer()` function (which is called when an AJAX request goes to the `wp-admin/admin-ajax.php` script) after the user’s login and password have been successfully validated from cookies, to allow a plugin to force WordPress to die for extra security reasons. Note that `check_ajax_referer` is also a “pluggable” function, meaning that plugins can override it; see Plugin API).

customize_controls_enqueue_scripts : triggered after the WP Theme Customizer after customize_controls_init was called, its actions/callbacks executed, and its own styles and scripts enqueued, so you can use this hook to register your own scripts and styles for WP Theme Customizer. For use with the Theme Customization API (as of Version 3.4).

customize_register : Runs on every request, allowing developers to register new theme options and controls for use with the Theme Customization API (as of Version 3.4).

customize_preview_init : Allows you to enqueue assets (such as javascript files) directly in the Theme Customizer only. For use with the Theme Customization API (as of Version 3.4).

deactivated_plugin : Runs any time any plugin is successfully de-activated

generate_rewrite_rules : Runs after the rewrite rules are generated. Action function arguments: `WP_Rewrite` object (`$wp_rewrite`) by reference. Note that it is easier to use the r`ewrite_rules_array` filter instead of this action, to modify the rewrite rules.

init : Runs after WordPress has finished loading but before any headers are sent. Useful for intercepting $_GET or $_POST triggers.

loop_end : Runs after the last post of the WordPress loop is processed.

loop_start : Runs before the first post of the WordPress loop is processed.

network_admin_menu : Runs when the basic menu structure is prepared for the Network administration page. (Administration Menus)

parse_query : Runs at the end of query parsing in the main query or any instance of WP_Query, such as query_posts, get_posts, or get_children. Action function arguments: `WP_Query` object by reference.

parse_request : Runs after the query request is parsed inside the main WordPress function `wp`. Action function argument: `WP` object ($wp) by reference.

pre_get_posts : Runs before a query is executed in the main query or any instance of `WP_Query`, such as `query_posts()`, `get_posts()`, or `get_children()`. This hook is called after the query variable object is created, but before the query is actually run, and can be used to to alter the primary query before it is run. Also see `is_main_query()`. Action function arguments: `WP_Query` object by reference.

sanitize_comment_cookies : Runs after cookies have been read from the HTTP request.

send_headers : Runs after the basic HTTP headers are sent inside the main WordPress function `wp()`. Action function argument: `WP` object ($wp) by reference.

shutdown : Runs when the page output is complete.

update_{$meta_type}_meta : Runs when a metadata gets saved.

updated_{$meta_type}_meta : Runs when a metadata has been updated.

upgrader_process_complete : Runs when the plugin downloader/upgrader class finishes running

wp_loaded : This hook is fired once WP, all plugins, and the theme are fully loaded and instantiated.

wp : Executes after the query has been parsed and post(s) loaded, but before any template execution, inside the main WordPress function `wp()`. Useful if you need to have access to post data but can't use templates for output. Action function argument: `WP` object (`$wp`) by reference.

Admin Login Actions

This section contains actions that the WordPress Admin login page uses to handle display, authentication, registering, resetting passwords, forgot password, and other user handling.

login_init : Fires when the login form is initialized.

login_form_{$action} : Fires before a specified login form action.

login_enqueue_scripts : Enqueue scripts and styles for the login page.

login_head : Fires in the login page header after scripts are enqueued.

login_header : Fires in the login page header after the body tag is opened.

login_form : Fires following the 'Password' field in the login form.

lostpassword_post : Fires before errors are returned from a password reset request.

admin_email_confirm : Fires before the admin email confirm form.

admin_email_confirm_form : Fires inside the admin-email-confirm-form form tags, before the hidden fields.

lost_password : Fires before the lost password form.

lostpassword_form : Fires inside the lostpassword form tags, before the hidden fields.

validate_password_reset : Fires before the password reset procedure is validated.

resetpass_form : Fires following the 'Strength indicator' meter in the user password reset form.

register_form : Fires following the 'Email' field in the user registration form.

user_request_action_confirmed : Fires an action hook when the account action has been confirmed by the user.

login_footer : Fires in the login page footer.

Further Reading

- Plugin Handbook – description of how to write a plugin
- Plugin API – article on how to use filters and actions
- Plugin API/Filter Reference – reference list for filter hooks
- Plugin Resources – comprehensive list of plugin-related resources
- Hooks Database – documentation for all hooks in the official WordPress Code Reference.
- WordPress Hooks Database, a database of all WordPress' hooks, showing which version they come from, and linking to the source code spots that use them

---

□□□ #2
□ □□□ □□□ 2026-10-31 18:08:17 UTC
□ □□□ □□□ 2026-10-31 18:08:19 UTC