

WORDPRESS HOOKS

Cron & Scheduling

Hooks in this category help you schedule recurring tasks, control cron spawning, and manage background cleanup in WordPress.



Handle schedules, cron requests and background jobs with these hooks.



cron_schedules

Registers custom cron interval schedules.



cron_request

Filters the HTTP request WordPress uses to spawn WP-Cron.



wp_scheduled_delete

Fires daily via WP-Cron, right before old trashed posts are purged.

4WP

4wp.dev/hooks/category/cron/

Better timing. Smarter automation.

Use these hooks to customize schedules, cron requests, and recurring cleanup in WordPress.

wp_scheduled_delete

The `wp_scheduled_delete` action hook is fired when WordPress runs a **scheduled delete** for a post, term, comment, or user. It's useful for cleanup tasks, logging, notifications, or removing related data when content is permanently deleted.

*Fired when a scheduled delete runs, so you can perform cleanup tasks after WordPress **permanently** deletes content.*



```
add_action( 'wp_scheduled_delete',  
'my_cleanup_related_data' );
```

Clean Up Related Data



```
add_action( 'wp_scheduled_delete',  
'my_log_deleted_item' );
```

Log Deleted Items



```
add_action( 'wp_scheduled_delete',  
'my_send_notification' );
```

Send a Notification



```
add_action( 'wp_scheduled_delete',  
'my_custom_delete_logic' );
```

Run Custom Logic

```
<?php  
add_action( 'wp_scheduled_delete',  
    'my_cleanup_related_data' );  
  
function my_cleanup_related_data( $post_id ) {  
    // Get post type before it's deleted.  
    $post = get_post( $post_id );  
    if ( ! $post ) {  
        return;  
    }  
  
    // Remove related data, files, or metadata.  
    // e.g. delete custom records or files.  
    // Your code here.  
}  
?>
```

cron_schedules

The **cron_schedules** filter lets you register custom recurring WP-Cron intervals.



Default schedules

WordPress core ships with hourly, twice daily, and daily schedules. Use this filter to add custom intervals such as every 15 minutes or weekly.



When it's used

The filter is read when `wp_get_schedules()` runs, typically before `wp_schedule_event()` registers a recurring event.



Common use cases

- Add a custom recurring interval
- Register weekly or non-standard intervals
- Match plugin task frequency
- Support finer-grained sync/polling



Common mistake

Scheduling an event with an interval name that was never registered through **cron_schedules**, so it never runs as expected.



PHP

```
<?php
add_filter( 'cron_schedules', function( $schedules ) {
    $schedules['every_fifteen_minutes'] = [
        'interval' => 15 * MINUTE_IN_SECONDS,
        'display'  => 'Every 15 Minutes',
    ];
    return $schedules;
} );
```

Extend what's possible.

Add custom WP-Cron intervals with the **cron_schedules** filter.

cron_request

The **cron_request** filter lets you modify the HTTP request WordPress uses to spawn WP-Cron.



Extend timeout

Adjust the timeout for slow cron jobs.



Add authentication

Attach headers to the internal cron request.



Log cron activity

Monitor when WordPress spawns WP-Cron.



Proxy / redirect

Adjust the request for proxies or load balancers.



Control the WP-Cron spawn request with the **<filter>** hook.

```
<?php
add_filter( 'cron_request',
function( $request ) {
    // Increase timeout for cron spawn
    $request['args']['timeout'] = 30;
    return $request;
} );
```



4wp.dev/hooks/filters/cron_request

Power your **<filter>**.

Control WordPress cron-spawn HTTP requests with the **cron_request** filter.