FAQ

- FAQ
  - Is it safe to release Action Scheduler in my plugin? Won't its functions conflict with another copy
  - I don't want to use WP-Cron. Does Action Scheduler depend on WP-Cron?
  - I don't want to use WP-Cron, ever. Does Action Scheduler replace WP-Cron?
  - How does Action Scheduler store its data?
  - Can I use a different storage scheme?
  - Can I use a different storage scheme just for logging?
  - I want to run Action Scheduler only on a dedicated application server in my cluster. Can I do that?
  - Is Action Scheduler safe to use on my production site?
  - How does Action Scheduler work on WordPress Multisite?
  - How can I change the Action Scheduler User-Agent to better identify its requests?
  - My site has past-due actions; what can I do?

FAQ

Is it safe to release Action Scheduler in my plugin? Won't its functions conflict with another copy of the library?

Action Scheduler is designed to be used and released in plugins. It avoids redeclaring public API functions when more than one copy of the library is being loaded by different plugins. It will also load only the most recent version of itself (by checking registered versions after all plugins are loaded on the `'plugins_loaded'` hook).

To use it in your plugin (or theme), simply require the `action-scheduler/action-scheduler.php` file. Action Scheduler will take care of the rest. **Note:** Action Scheduler is only loaded from a theme if it is not included in any active plugins.

I don't want to use WP-Cron. Does Action Scheduler depend on WP-Cron?

By default, Action Scheduler is initiated by WP-Cron (and the `'shutdown'` hook on admin requests). However, it has no dependency on the WP-Cron system. You can initiate the Action Scheduler queue in other ways with just one or two lines of code.

For example, you can start a queue directly by calling:

```
ActionScheduler::runner()->run();
```

Or trigger the `'action_scheduler_run_queue'` hook and let Action Scheduler do it for you:

```
do_action( 'action_scheduler_run_queue', $context_identifier );
```

Further customization can be done by extending the `ActionScheduler_Abstract_QueueRunner` class to create a custom Queue Runner. For an example of a customized queue runner, see the `ActionScheduler_WPCLI_QueueRunner`, which is used when running WP CLI.

Want to create some other method for initiating Action Scheduler? Open a new issue, we'd love to help you with it.

I don't want to use WP-Cron, ever. Does Action Scheduler replace WP-Cron?

By default, Action Scheduler is designed to work alongside WP-Cron and not change any of its behaviour. This helps avoid unexpectedly overriding WP-Cron on sites installing your plugin, which may have nothing to do with WP-Cron.

However, we can understand why you might want to replace WP-Cron completely in environments within your control, especially as it gets you the advantages of Action Scheduler. This should be possible without too much code.

You could use the `'schedule_event'` hook in WordPress to use Action Scheduler for only newly scheduled WP-Cron jobs and map the `$event` param to Action Scheduler API functions.

Alternatively, you can use a combination of the `'pre_update_option_cron'` and `'pre_option_cron'` hooks to override all new and previously scheduled WP-Cron jobs (similar to the way Cavalcade does it).

If you'd like to create a plugin to do this automatically and want to share your work with others, open a new issue to let us know, we'd love to help you with it.

How does Action Scheduler store its data?

Action Scheduler 3.0 and newer stores data in custom tables prefixed with `actionscheduler_`. For the list of all tables and their schemas, refer to the `ActionScheduler_StoreSchema` class.

Prior to Action 3.0, actions were a custom post type, and data was stored in `wp_posts`, `wp_postmeta` and related tables.

Action Scheduler 3+ migrates data from the custom post type to custom tables.

Can I use a different storage scheme?

Of course! Action Scheduler data storage is completely swappable, and always has been.

If you choose to, you can actually store them anywhere, like in a remote storage service from Amazon Web Services.

To implement a custom store:

1. extend the abstract `ActionScheduler_Store` class, being careful to implement each of its methods
2. attach a callback to `'action_scheduler_store_class'` to tell Action Scheduler your class is the one which should be used to manage storage, e.g.

```
function eg_define_custom_store( $existing_storage_class ) {
	return 'My_Radical_Action_Scheduler_Store';
}
add_filter( 'action_scheduler_store_class', 'eg_define_custom_store', 10, 1 );
```

Take a look at the `classes/data-stores/ActionScheduler_DBStore.php` class for an example implementation of `ActionScheduler_Store`.

If you'd like to create a plugin to do this automatically and release it publicly to help others, open a new issue to let us know, we'd love to help you with it.

Can I use a different storage scheme just for logging?

Of course! Action Scheduler's logger is completely swappable, and always has been. You can also customise where logs are stored, and the storage mechanism.

To implement a custom logger:

1. extend the abstract `ActionScheduler_Logger` class, being careful to implement each of its methods
2. attach a callback to `'action_scheduler_logger_class'` to tell Action Scheduler your class is the one which should be used to manage logging, e.g.

```
function eg_define_custom_logger( $existing_storage_class ) {
	return 'My_Radical_Action_Scheduler_Logger';
}
add_filter( 'action_scheduler_logger_class', 'eg_define_custom_logger', 10, 1 );
```

Take a look at the `classes/data-stores/ActionScheduler_DBLogger.php` class for an example implementation of `ActionScheduler_Logger`.

I want to run Action Scheduler only on a dedicated application server in my cluster. Can I do that?

Wow, now you're really asking the tough questions. In theory, yes, this is possible. The `ActionScheduler_QueueRunner` class, which is responsible for running queues, is swappable via the `'action_scheduler_queue_runner_class'` filter.

Because of this, you can effectively customise queue running however you need. Whether that means tweaking minor things, like not using WP-Cron at all to initiate queues by overriding `ActionScheduler_QueueRunner::init()`, or completely changing how and where queues are run, by overriding `ActionScheduler_QueueRunner::run()`.

Is Action Scheduler safe to use on my production site?

Yes, absolutely! Action Scheduler is actively used on tens of thousands of production sites already. Right now it's responsible for scheduling everything from emails to payments.

In fact, every month, Action Scheduler processes millions of payments as part of the WooCommerce Subscriptions extension.

It requires no setup, and won't override any WordPress APIs (unless you want it to).

How does Action Scheduler work on WordPress Multisite?

Action Scheduler is designed to manage the scheduled actions on a single site. It has no special handling for running queues across multiple sites in a multisite network. That said, because its storage and Queue Runner are completely swappable, it would be possible to write multisite handling classes to use with it.

If you'd like to create a multisite plugin to do this and release it publicly to help others, open a new issue to let us know, we'd love to help you with it.

How can I change the Action Scheduler User-Agent to better identify its requests?

Action Scheduler has a filter available named `as_async_request_queue_runner_post_args` which can be used to filter the arguments that are being sent to the `wp_remote_post` call.

The User-Agent parameter is just one of them and can be adjusted as follows:

```
function eg_define_custom_user_agent( $args ) {
	$versions           = ActionScheduler_Versions::instance();
	$args['user-agent'] = 'Action Scheduler/' . $versions->latest_version();

	return $args;
}
add_filter( 'as_async_request_queue_runner_post_args', 'eg_define_custom_user_agent', 10, 1 );
```

My site has past-due actions; what can I do?

Actions that are past-due have missed their scheduled date; because of how WP Cron works, it is normal to have some past-due actions.

If there are several past-due actions more than one day old, there may be something wrong with your site. If you are using WooCommerce and need help determining the issue, please contact us on our forum page.

□□□ #1
□ □□□ □□□ 2026-08-29 22:03:34 UTC
□ □□□ □□□ 2026-08-29 22:03:34 UTC