API Reference

- WP-Cron APIs vs. Action Scheduler APIs
- API Function Availability
- Function Reference / as_enqueue_async_action()
  - Description
  - Usage
  - Parameters
  - Return value
- Function Reference / as_schedule_single_action()
  - Description
  - Usage
  - Parameters
  - Return value
- Function Reference / as_schedule_recurring_action()
  - Description
  - Usage
  - Parameters
  - Return value
- Function Reference / as_schedule_cron_action()
  - Description
  - Usage
  - Parameters
  - Return value
- Function Reference / as_unschedule_action()
  - Description
  - Usage
  - Parameters
  - Return value
- Function Reference / as_unschedule_all_actions()
  - Description

    - Usage
    - Parameters
    - Return value
- Function Reference / as_next_scheduled_action()
    - Description
    - Usage
    - Parameters
    - Return value
- Function Reference / as_has_scheduled_action()
    - Description
    - Usage
    - Parameters
    - Return value
- Function Reference / as_get_scheduled_actions()
    - Description
    - Usage
    - Parameters
    - Return value
- Function Reference / as_supports()
    - Description
    - Usage
    - Parameters
    - Return value
    - Example

Action Scheduler provides a range of functions for scheduling hooks to run at some time in the future on one or more occasions.

To understand the scheduling functions, it can help to think of them as extensions to WordPress' `do_action()` function that add the ability to delay and repeat when the hook will be triggered.

WP-Cron APIs vs. Action Scheduler APIs

The Action Scheduler API functions are designed to mirror the WordPress WP-Cron API functions.

Functions return similar values and accept similar arguments to their WP-Cron counterparts. The notable differences are:

- `as_schedule_single_action()` & `as_schedule_recurring_action()` will return the ID of the scheduled action rather than boolean indicating whether the event was scheduled
- `as_schedule_recurring_action()` takes an interval in seconds as the recurring interval rather than an arbitrary string
- `as_schedule_single_action()` & `as_schedule_recurring_action()` can accept a `$group` parameter to group different actions for the one plugin together.
- the `wp_` prefix is substituted with `as_` and the term `event` is replaced with `action`

API Function Availability

As mentioned in the Usage - Load Order section, Action Scheduler will initialize itself on the `'init'` hook with priority `1`. While API functions are loaded prior to this and can be called, they should not be called until after `'init'` with priority `1`, because each component, like the data store, has not yet been initialized.

Do not use Action Scheduler API functions prior to `'init'` hook with priority `1`. Doing so could lead to unexpected results, like data being stored in the incorrect location. To make this easier:

- Action Scheduler provides `Action_Scheduler::is_initialized()` for use in hooks to confirm that the data stores have been initialized.
- It also provides the `'action_scheduler_init'` action hook. It is safe to call API functions during or after this event has fired (tip: you can take advantage of WordPress's did_action() function to check this).

Function Reference / `as_enqueue_async_action()`

Description

Enqueue an action to run one time, as soon as possible.

Usage

```
as_enqueue_async_action( $hook, $args, $group, $unique, $priority );
```

Parameters

- **$hook** (string)(required) Name of the action hook.
- **$args** (array) Arguments to pass to callbacks when the hook triggers. Default: `array()`.
- **$group** (string) The group to assign this job to. Default: ''.
- **$unique** (boolean) Whether the action should be unique. Default: `false`.
- **$priority** (integer) Lower values take precedence over higher values. Defaults to 10, with acceptable values falling in the range 0-255.

Return value

`(integer)` the action's ID. Zero if there was an error scheduling the action. The error will be sent to `error_log`.

Function Reference / `as_schedule_single_action()`

Description

Schedule an action to run one time at some defined point in the future.

Usage

```
as_schedule_single_action( $timestamp, $hook, $args, $group, $unique, $priority );
```

Parameters

- **$timestamp** (integer)(required) The Unix timestamp representing the date you want the action to run.
- **$hook** (string)(required) Name of the action hook.
- **$args** (array) Arguments to pass to callbacks when the hook triggers. Default: `array()`.
- **$group** (string) The group to assign this job to. Default: ''.
- **$unique** (boolean) Whether the action should be unique. Default: `false`.

- **$priority** (integer) Lower values take precedence over higher values. Defaults to 10, with acceptable values falling in the range 0-255.)

Return value

`(integer)` the action's ID. Zero if there was an error scheduling the action. The error will be sent to `error_log`.

Function Reference / `as_schedule_recurring_action()`

Description

Schedule an action to run repeatedly with a specified interval in seconds.

Usage

```
as_schedule_recurring_action( $timestamp, $interval_in_seconds, $hook, $args, $group, $unique, $priority );
```

Parameters

- **$timestamp** (integer)(required) The Unix timestamp representing the date you want the action to run.
- **$interval_in_seconds** (integer)(required) How long to wait between runs.
- **$hook** (string)(required) Name of the action hook.
- **$args** (array) Arguments to pass to callbacks when the hook triggers. Default: `array()`.
- **$group** (string) The group to assign this job to. Default: ''.
- **$unique** (boolean) Whether the action should be unique. Default: `false`.
- **$priority** (integer) Lower values take precedence over higher values. Defaults to 10, with acceptable values falling in the range 0-255.

Return value

`(integer)` the action's ID. Zero if there was an error scheduling the action. The error will be sent to `error_log`.

Function Reference / `as_schedule_cron_action()`

Description

Schedule an action that recurs on a cron-like schedule.

If execution of a cron-like action is delayed, the next attempt will still be scheduled according to the provided cron expression.

Usage

```
as_schedule_cron_action( $timestamp, $schedule, $hook, $args, $group, $unique, $priority );
```

Parameters

- **$timestamp** (integer)(required) The Unix timestamp representing the date you want the action to run.
- **$schedule** (string)(required) $schedule A cron-like schedule string, see http://en.wikipedia.org/wiki/Cron.
- **$hook** (string)(required) Name of the action hook.
- **$args** (array) Arguments to pass to callbacks when the hook triggers. Default: `array()`.
- **$group** (string) The group to assign this job to. Default: ''.
- **$unique** (boolean) Whether the action should be unique. Default: `false`.
- **$priority** (integer) Lower values take precedence over higher values. Defaults to 10, with acceptable values falling in the range 0-255.

Return value

`(integer)` the action's ID. Zero if there was an error scheduling the action. The error will be sent to `error_log`.

Function Reference / `as_unschedule_action()`

Description

Cancel the next occurrence of a scheduled action.

Usage

```
as_unschedule_action( $hook, $args, $group );
```

Parameters

- **$hook** (string)(required) Name of the action hook.
- **$args** (array) Arguments passed to callbacks when the hook triggers. Default: `array()`.
- **$group** (string) The group the job is assigned to. Default: ''.

Return value

`(null)`

Function Reference / `as_unschedule_all_actions()`

Description

Cancel all occurrences of a scheduled action.

Usage

```
as_unschedule_all_actions( $hook, $args, $group )
```

Parameters

- **$hook** (string)(required) Name of the action hook.
- **$args** (array) Arguments passed to callbacks when the hook triggers. Default: `array()`.
- **$group** (string) The group the job is assigned to. Default: ''.

Return value

`(string|null)` The scheduled action ID if a scheduled action was found, or null if no matching action found.

Function Reference / `as_next_scheduled_action()`

Description

Returns the next timestamp for a scheduled action.

Usage

```
as_next_scheduled_action( $hook, $args, $group );
```

Parameters

- **$hook** (string)(required) Name of the action hook. Default: *none*.
- **$args** (array) Arguments passed to callbacks when the hook triggers. Default: `array()`.
- **$group** (string) The group the job is assigned to. Default: ''.

Return value

`(integer|boolean)` The timestamp for the next occurrence of a pending scheduled action, true for an async or in-progress action or false if there is no matching action.

Function Reference / `as_has_scheduled_action()`

Description

Check if there is a scheduled action in the queue, but more efficiently than as_next_scheduled_action(). It's recommended to use this function when you need to know whether a specific action is currently scheduled. *Available since 3.3.0.*

Usage

```
as_has_scheduled_action( $hook, $args, $group );
```

Parameters

- **$hook** (string)(required) Name of the action hook. Default: *none*.
- **$args** (array) Arguments passed to callbacks when the hook triggers. Default: `array()`.
- **$group** (string) The group the job is assigned to. Default: ''.

Return value

`(boolean)` True if a matching action is pending or in-progress, false otherwise.

Function Reference / `as_get_scheduled_actions()`

Description

Find scheduled actions.

Usage

```
as_get_scheduled_actions( $args, $return_format );
```

Parameters

- **$args** (array) Arguments to search and filter results by. Possible arguments, with their default values:
  - `'hook' => ''` - the name of the action that will be triggered
  - `'args' => NULL` - the args array that will be passed with the action
  - `'date' => NULL` - the scheduled date of the action. Expects a DateTime object, a unix timestamp, or a string that can parsed with strtotime().
  - `'date_compare' => '<=` ' - operator for testing "date". accepted values are '!=', '>', '>=', '<', '<=', '='
  - `'modified' => NULL` - the date the action was last updated. Expects a DateTime object, a unix timestamp, or a string that can parsed with strtotime().
  - `'modified_compare' => '<='` - operator for testing "modified". accepted values are '!=', '>', '>=', '<', '<=', '='
  - `'group' => ''` - the group the action belongs to
  - `'status' => ''` - ActionScheduler_Store::STATUS_COMPLETE or ActionScheduler_Store::STATUS_PENDING
  - `'claimed' => NULL` - TRUE to find claimed actions, FALSE to find unclaimed actions, a string to find a specific claim ID
  - `'per_page' => 5` - Number of results to return
  - `'offset' => 0`
  - `'orderby' => 'date'` - accepted values are 'hook', 'group', 'modified', or 'date'
  - `'order' => 'ASC'`
- **$return_format** (string) The format in which to return the scheduled actions: 'OBJECT', 'ARRAY_A', or 'ids'. Default: '*OBJECT*'.

Return value

`(array)` Array of action rows matching the criteria specified with `$args`.

Function Reference / `as_supports()`

Description

Check if a specific feature is supported by the current version of Action Scheduler.

Available since 3.9.3.

Usage

```
as_supports( $feature );
```

Parameters

$feature (string) (required)
The feature to check support for.
Currently supported:

- `'ensure_recurring_actions_hook'` — Indicates support for the `action_scheduler_ensure_recurring_actions` hook.

Return value

`(boolean)`
True if the feature is supported, false otherwise.

Example

```
if ( as_supports( 'ensure_recurring_actions_hook' ) ) {
    // Safe to depend on the 'action_scheduler_ensure_recurring_actions' hook.
    add_action( 'action_scheduler_ensure_recurring_actions',
'my_plugin_schedule_my_recurring_action' );
}
```

□□□ #2
□ □□□ □□□ 2026-08-29 22:03:34 UTC
□ □□□ □□□ 2026-08-29 22:03:34 UTC