

Plugin Development: Scheduled Tasks

If your plugin needs to perform a recurring task, you can set up a scheduled task to run on your chosen schedule for the plugin.







Scheduled tasks let you run a function on a recurring basis at an interval of your choice. Example cases where this is useful:

- Check for tickets from an external source (our Facebook and X channels)
- Perform actions that are not currently covered by our macro, follow up or escalation rules systems

A task can be registered by using the Scheduled Task model's register method in the plugin's activate function. The first parameter is the name of the task, the second is a description that is shown on the scheduled tasks page and the third is the frequency in seconds. The frequency should be reduced by 15 seconds as this is the buffer we account for to ensure it does not add an extra minute when actually running. For example, you would enter 885 for every 15 minutes, or 3585 for every hour.

```
public function activate(){ // Add scheduled task  AppModule
CoreModelsScheduledTask::register(    'My Task',
    'This is my task for My Plugin.',    885  );  return true;}
```

When the plugin is now activated, it will add the scheduled task which will be visible when you visit Settings -> Core -> Scheduled Tasks.

Handle inactive tickets	Sends waiting for response emails and closes tickets that have become inactive with no follow up from the user.	Every 1 hour	5 minutes ago	 
My Task	This is my task for My Plugin	Every 15 minutes	Never	 
Process automatic ticket macros	Checks if any automatic macros now fit the conditions on non-resolved tickets (on which it has not previous ran) and runs them.	Every 15 minutes	5 minutes ago	 

A task can be removed by using the Scheduled Task model's deregister method in the plugin's deactivate and uninstall functions. It should be ran in both functions as the task must be removed if the plugin is no longer active or installed.

```
public function deactivate(){
    // Remove scheduled task AppModulesCoreModels
    ScheduledTask::deregister(); return true;}
public function uninstall(){
    // Remove scheduled task AppModulesCoreModels
    ScheduledTask::deregister(); return true;}
```

The scheduled task is now set up and scheduled to run on the interval you've set, but you still need to code what it should do. The plugin controller must implement ScheduledTaskInterface and then define a runTask function which is what is ran each time the scheduled task runs, like below.

Controllers/HelloWorld.php

```
<?php namespace AddonsPluginsHelloWorldControllers;
use AppModulesCoreControllersPluginsPlugin;
use AppModulesCoreInterfacesScheduledTaskInterface;
class HelloWorld extends Plugin implements ScheduledTaskInterface{
    ... public function runTask() {
        // Run our task code    }}
```

The function doesn't need a return statement, but it should throw an exception with a message if there is an error, this will log the error in application log files.

Online URL:

<https://docs.supportpro.vn/article/plugin-development-scheduled-tasks-225.html>