CRON SCHEDULER PRO

User Guide

Cron tasks in Magento

Cron Scheduler Pro is an extension that is solely available for Magento 2 and freely inspired from the excellent *AOE Scheduler*.

Cron Scheduler Pro makes the cron tasks easier and more intuitive to manage. **However, this extension doesn't dispense to configure the main cron task from the server side.** For more information about how to configure the main cron task at the server level, please refer to the official Magento guide.

The cron tasks mechanism in Magento requires to understand a few basic principles:

1. The main cron task that runs on the server triggers regularly (ideally each minute).

```
bin/magento cron:run
```

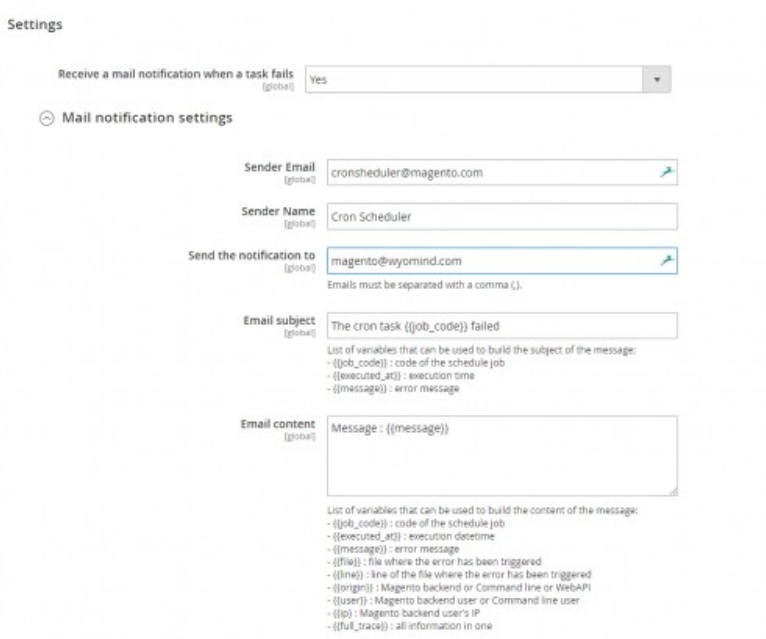
2. The above command analyses each cron job (configured in the different enabled modules) and for each one of them, it programs the cron tasks for a few hours ahead depending on your preferences in:
   STORE › CONFIGURATION › SYSTEM › CRON
3. Then, the same command runs the cron tasks that match with the current time, flags the tasks that are too old as missed and cleans up the tasks history.

Configure Cron Scheduler in a general way

Cron scheduler only requires a few settings.

Email notification

Define whether or not you want to receive an email alert when a cron task fails.



When the option is enabled, you have to fill in the below inputs.

Sender Email

Email of the sender (the email address doesn't require to be a valid/existing email address).

Example: cronsheduler@magento.com

Sender Name

Name of the sender.

Example: Cron Scheduler

Send the notification to

Email address of the recipient (each address must be separated by a comma).

Example: wyomind@email1.com,wyomind@email2.com

Email subject

Subject of the email.

You can use the following placeholders:

- **{{job_code}}**
  Code of the cron job.
- **{{executed_at}}**:
  Date/time of the task (yyyy-mm-dd hh::mm::ss).
- **{{message}}**
  Error message that is generated by the cron task.

Email content

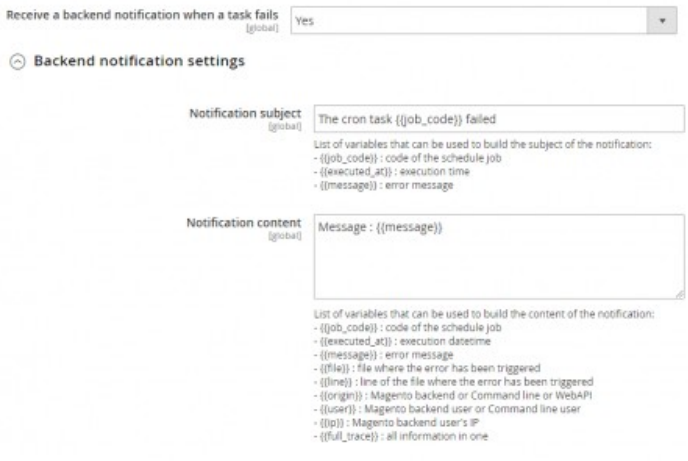
Content of the message.

You can use the following placeholders:

- **{{job_code}}**
  Code of the cron job.
- **{{executed_at}}**
  Date/time of the task (yyyy-mm-dd hh::mm::ss).
- **{{message}}**
  Error message that is generated by the cron task.
- **{{file}}**
  File where the error happens.
- **{{line}}**
  Line number in the file where the error happens.
- **{{origin}}**
  Magento backoffice or CLI or API.
- **{{user}}**
  User's name (for admin users or user's name of the API) or CLI.
- **{{ip}}**
  User's IP.
- **{{full_trace}}**
  Shortcut for all above placeholders.

Backoffice notification

Define whether or not you want to receive a notification in your Magento backoffice when a cron task fails.



When the option is enabled, you have to fill the below inputs.

Notification subject

Subject of the notification

You can use the following placeholders:

- **{{job_code}}**
  Code of the cron job.
- **{{executed_at}}**
  Date/time of the task (yyyy-mm-dd hh::mm::ss).
- **{{message}}**
  Error message that is generated by the cron task.

Notification content

Content of the notification.

You can use the following placeholders:

- **{{job_code}}**
  Code of the cron job.
- **{{executed_at}}**
  Date/time of the task (yyyy-mm-dd hh::mm::ss).
- **{{message}}**
  Error message that is generated by the cron task.
- **{{file}}**
  File where the error happens.
- **{{line}}**
  Line number in the file where the error happens.
- **{{origin}}**
  Magento backoffice or CLI or API.
- **{{user}}**
  User's name (for admin users or user's name of the API) or CLI.
- **{{ip}}**
  User's IP.
- **{{full_trace}}**
  Shortcut for all above placeholders.

Cron Scheduler views

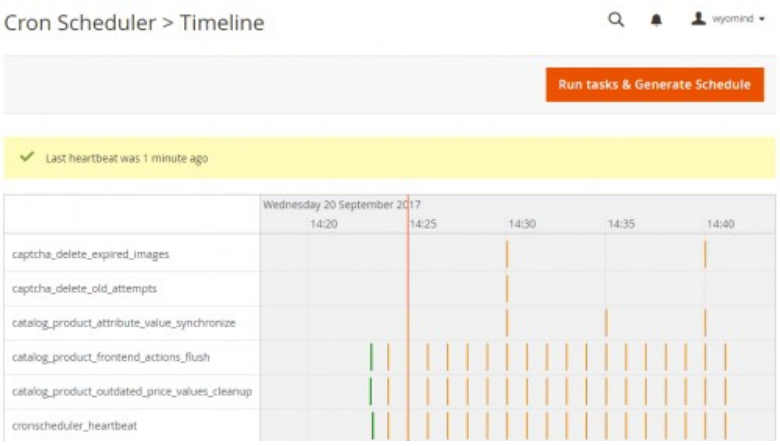
Cron Scheduler offers three distinct views allowing to visualize and manage the cron tasks.

- **Tasks Timeline**
  Chronological view of the executed and scheduled cron tasks.
- **Task list**
  Grid view of the executed and scheduled cron tasks.
- **Job configuration**
  Grid view of all cron jobs that generate new cron tasks.

Tasks Timeline

To display the timeline, go to:
SYSTEM › CRON SCHEDULER › TASKS TIMELINE

The timeline shows all the cron tasks executed and scheduled.



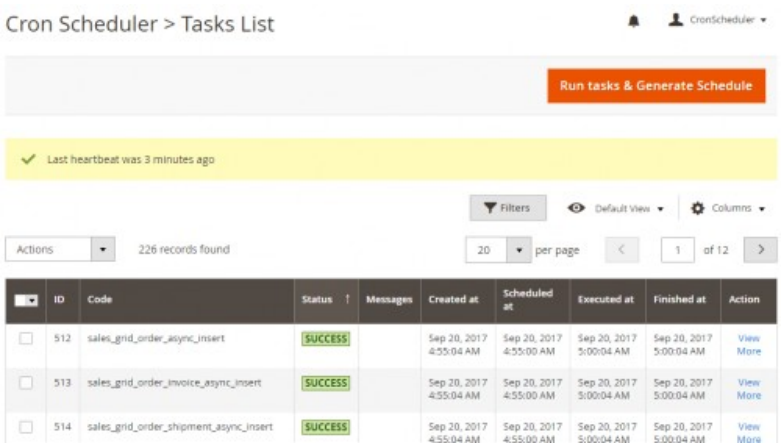
You can get more information by moving the mouse over each task mark.

Zoom in and out using the mouse wheel on the timeline.

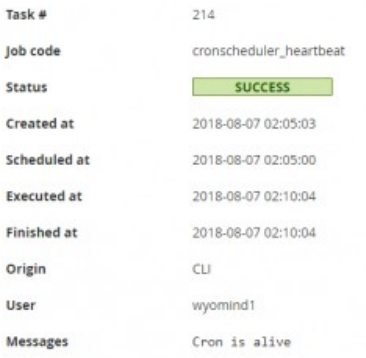
Tasks List

To display the tasks list, go to:
SYSTEM › CRON SCHEDULER › TASKS LIST

The grid lists all the cron tasks executed and scheduled.



You can get more information by clicking on **View more** on the right column.

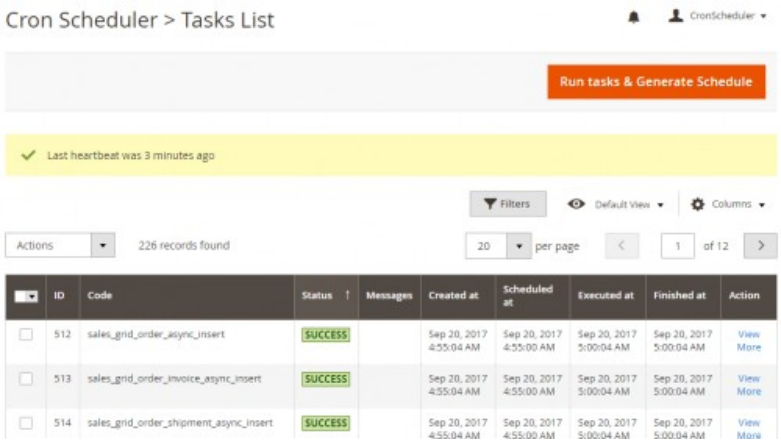


You can also delete the cron tasks by ticking the checkboxes and by clicking on **Delete** in the mass action drop-down.

Jobs Configuration

To display the jobs configuration list, go to:
SYSTEM CRON SCHEDULER JOBS CONFIGURATION

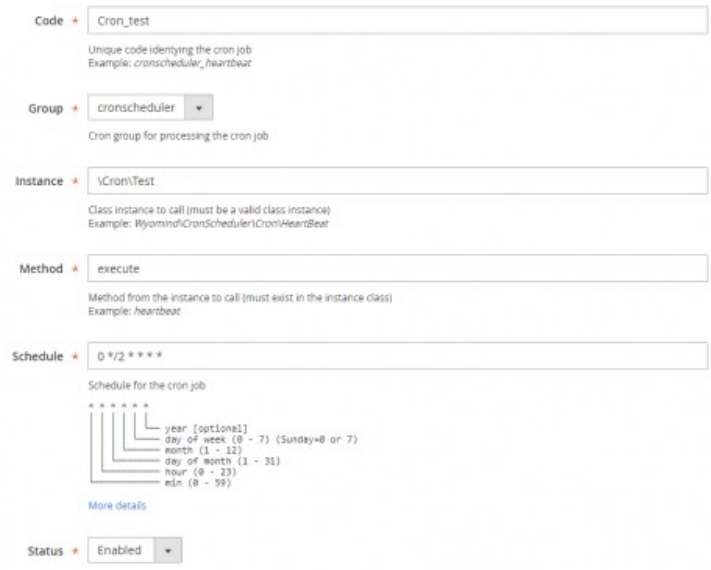
This grid lists all the cron jobs that generate new cron tasks.



Create a new cron job

Attention, the cron job configuration is for Magento 2 advanced users.
Vous can create new cron jobs thanks to *Cron Scheduler Pro* without having to create new modules.

Click on .



Then fill the form:

- **Code**
  The unique code of the cron job.
- **Group**
  The group to which belongs the new job.
- **Instance**
  The PHP class to be used.
- **Method**
  The PHP method to be used from the PHP class.
- **Frequency**
  The frequency of the cron tasks to generate.
  The frequency must be written as a cron expression.
- **Status**
  The status of the cron job, enabled or disabled.
  A disabled cron job does not generate any cron tasks.

Save the new job by clicking on .

- The new cron job appears with a user icon **.**
- The system cron jobs appear with a cogwheel icon **.**

Example of a custom cron job in *app/code/Cron.php*

```
<?php
 namespace Cron;
 class Test {
 public function execute($schedule) {
 $om = \Magento\Framework\App\ObjectManager::getInstance();
 try{
 // YOUR LOGIC HERE
 $message="Test processus has run!";
 }
 catch(\exception $e){
 $message=$e->getMessage();
 }
 $schedule->setMessages($message);
 $schedule->save();
```

```
 }
}
```


Edit a cron job

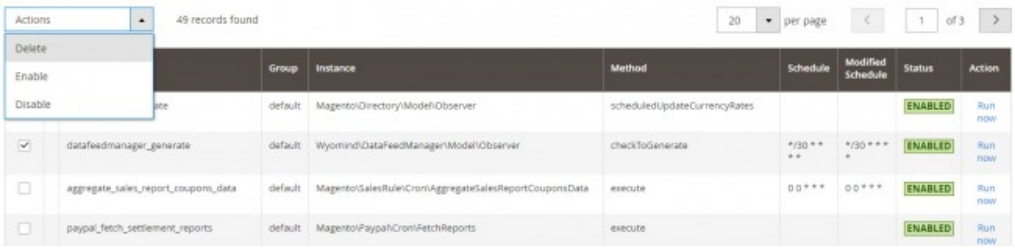
Attention, the cron job configuration is for Magento 2 advanced users.
To edit a cron job, you only need to click on the line, then the line becomes editable.

You can then update the following inputs:

- **Group**
- **Instance**
- **Method**
- **Frequency**
- **Status**

Delete a cron job

You can delete a custom cron job 👤 at any time but not the system cron jobs ⚙.



The system cron jobs must be disabled if you want don't want to generate cron tasks.

Run tasks & Generate schedule with Cron Scheduler

From the back-office

At any time you can run all cron jobs and generate new tasks from your back office, exactly like the below command line does:

```
bin/magento cron:run
```

You only have to click on from:
SYSTEM › CRON SCHEDULER › JOBS CONFIGURATION

You can also run a particular cron job individually by clicking on **run now** on the right column of the Jobs Configuration view.

From the CLI

Cron Scheduler includes new command lines that allow managing the cron jobs from the CLI.

- List all the cron tasks:

```
wyomind:cronscheduler:task:list
```

- Show details about a given cron task:

```
wyomind:cronscheduler:task:show task_id
```

- List all cron jobs:

```
wyomind:cronscheduler:job:list
```

- Run a specific cron job:

```
wyomind:cronscheduler:run job_code
```

From the API

Cron Scheduler also includes an API that allows running each cron job individually.

Below is an example of the API usage with the SOAP library:

```
<html>
 <head>
  <title>Web API call samples for Cron Scheduler Pro</title>
 </head>
```

```
<body>
 <?php
   if (!file_exists(__DIR__ . '/app/bootstrap.php')) {
   echo "The sample file must be placed in the Magento root folder!";
   return;
   }
   require __DIR__ . '/app/bootstrap.php';
   $bootstrap = \Magento\Framework\App\Bootstrap::create(BP, $_SERVER);
   $app = $bootstrap->createApplication('Magento\Framework\App\Http');
   $login = "LOGIN";
   $password = "PASSWORD";
   $consumerKey = 'd8durcngtxwf52nle30nx63vp5q6vl1o';
   $consumerSecret = 'ikoc08swj3odna2tytqmn91s69d2e0gw';
   $accessTokenSecret = 'yupmvtaqij6yfr9v47sefiwv91ve5qip';
   $accessToken = "g3rfdy7wuhujbhutthyofi9m4fe4pajg";
   $website = "http://www.website.com";
   $cronJob = "cronscheduler_heartbeat";
   // $cronJob = "dynamiccategory_reindex_all";
   echo "<pre>";


   /*************************************************************************
   * Token based authentication
   *************************************************************************/
   echo " .--------------------------------------------------------------------------.\n";
   echo " |  Token based authentication                                              |\n";
   echo " '--------------------------------------------------------------------------'\n";
   echo "\n";
   /**
   * SOAP V2 API
   */
   echo " == Executing the job '".$cronJob."' using the Soap v2 web API\n";
   echo "\n";
   $opts = ["http" => ["header" => "Authorization: Bearer " . $accessToken]];
   $context = stream_context_create($opts);
   $wsdlService = $website . "/index.php/soap/default?wsdl&services=wyomindCronSchedulerProCronV1";
   $soapClient = new \Zend\Soap\Client($wsdlService);
   $soapClient->setSoapVersion(SOAP_1_2);
   $soapClient->setStreamContext($context);
   // reun the job 'cronscehduler_heartbeat'
   $soapResult = $soapClient->wyomindCronSchedulerProCronV1run(array("jobCode" => $cronJob));
   if ($soapResult) {
   $result = json_decode($soapResult->result);
   echo "  >> Raw result: ".$soapResult->result."\n";
   if ($result->error) {
   echo "  >> Error when running the '".$cronJob."' job.\n";
   echo "  >> Message: " . $result->message;
   } else {
   echo "  >> The '".$cronJob."' job has been successfully executed.\n";
   }
   }
   echo "\n\n";



   ?>
 </body>
</html>
```