Truckmilk Manual

Jobs executor

June 2019

Sommaire

[1. Introduction 4](#_Toc22830594)

[2. Usage 4](#_Toc22830595)

[3. Installation 4](#_Toc22830596)

[3.1. Get the resource 4](#_Toc22830597)

[3.2. Install the page 4](#_Toc22830598)

[3.3. Install the Quartz 4](#_Toc22830599)

[3.4. Force a complete installation 5](#_Toc22830600)

[4. De installation 6](#_Toc22830601)

[4.1. Stop the Quartz job 6](#_Toc22830602)

[4.2. Remove the page 6](#_Toc22830603)

[4.3. Advanced setup 6](#_Toc22830604)

[5. Use jobs 6](#_Toc22830605)

[5.1. Create a job 6](#_Toc22830606)

[5.2. Schedule 7](#_Toc22830607)

[5.3. Host restrictions 8](#_Toc22830608)

[5.4. Parameters 8](#_Toc22830609)

[5.4.1. Document Input parameter 10](#_Toc22830610)

[5.4.2. Test parameters 11](#_Toc22830611)

[5.5. Execute it once 11](#_Toc22830612)

[5.6. Activate / Deactivate 12](#_Toc22830613)

[5.7. Access the report 12](#_Toc22830614)

[6. Embeded plug in 13](#_Toc22830615)

[6.1. Delete case 13](#_Toc22830616)

[6.2. Email User tasks 13](#_Toc22830617)

[6.3. Ping 14](#_Toc22830618)

[6.4. Purge Archived Case 14](#_Toc22830619)

[6.5. Purge Archived Case : Get List 14](#_Toc22830620)

[6.6. Purge Archived Case: Purge From List 14](#_Toc22830621)

[6.7. Replay Failed Tasks 14](#_Toc22830622)

[6.8. SLA 15](#_Toc22830623)

[7. Maintenance 15](#_Toc22830624)

[7.1. Information 15](#_Toc22830625)

[8. Build a new plug in 16](#_Toc22830626)

Content

# Introduction

This document explains how to use the Truckmilk Page.

# Usage

The page may be use in different context:

# Installation

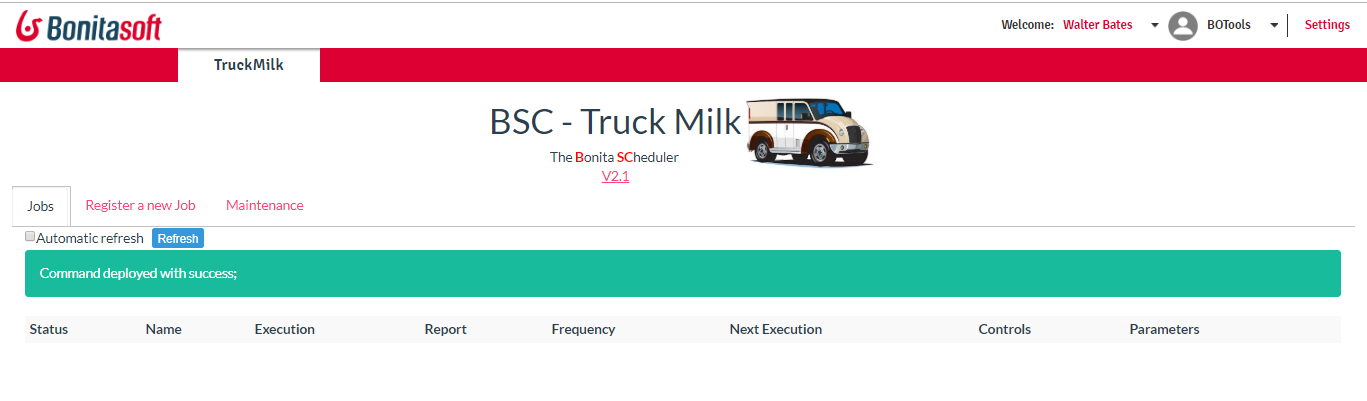
## Get the resource

Download the page from the community,

## Install the page

Then install the page as a Resource and reference it in a Profile or Application.

Access the page.



The first access, you should see a banner to explain the command is deployed with success

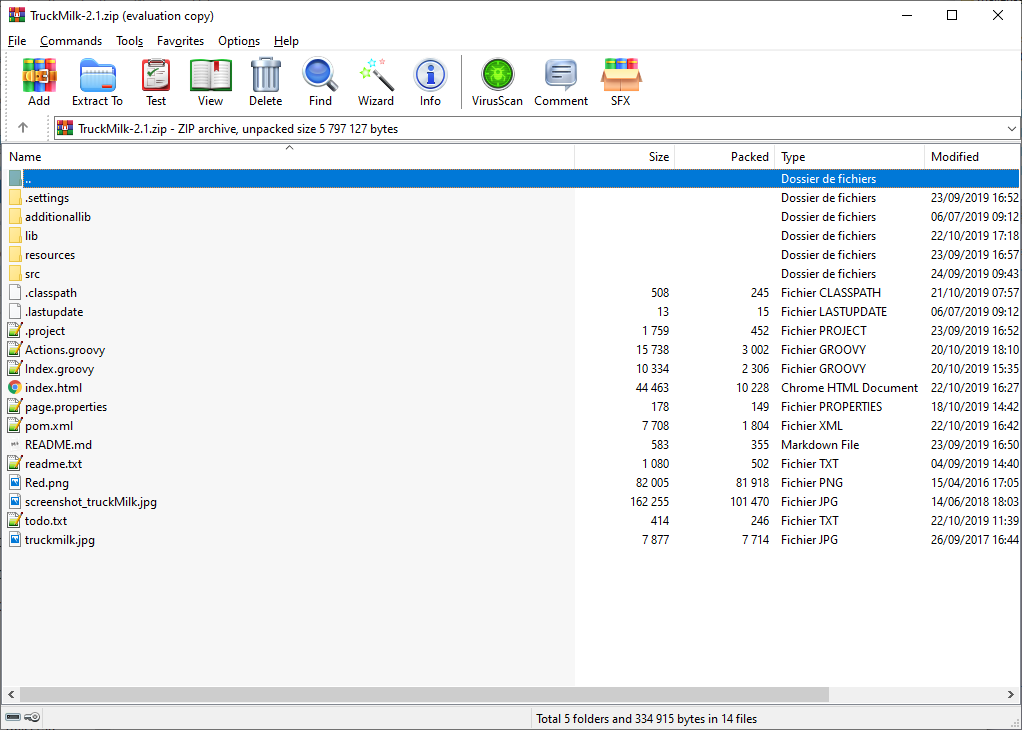
A Bonita Command is a JAVA Class deployed on the server. This allow the controller to check and start new jobs even if user is disconnected.

On a Cluster environment, each node will install the command. Command is saved in the database, and is backup when you back up the Bonita database

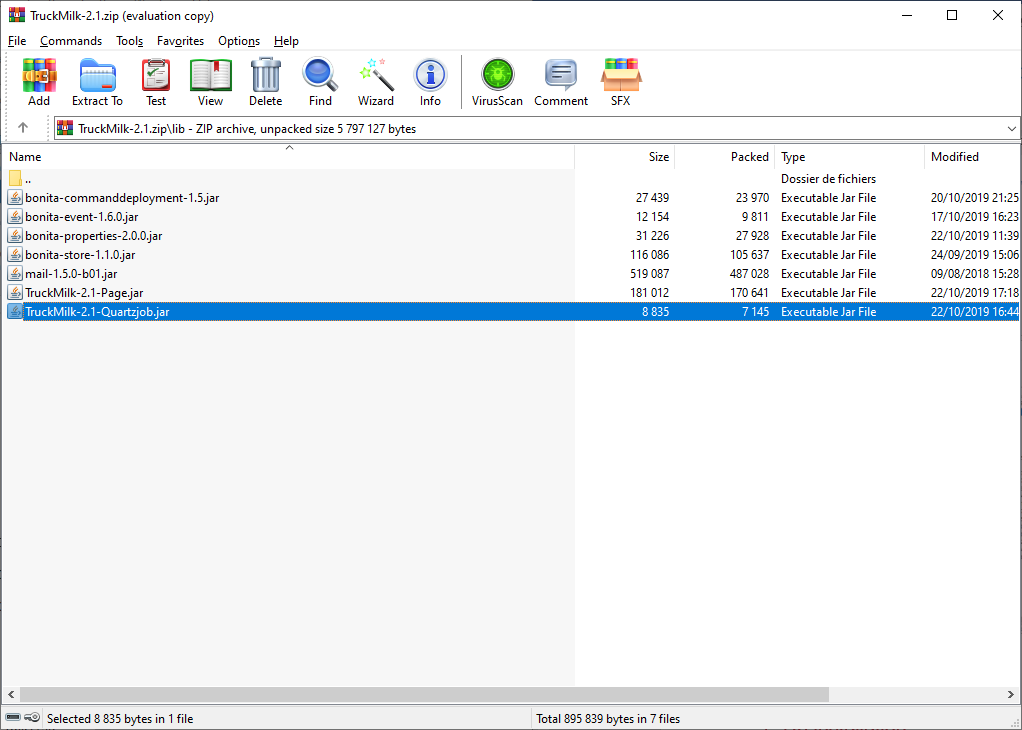
The second step consist to install the Quartz Job JAR file.

## Install the Quartz

Unzip the Truck milk ZIP file.



Access the lib directory



Copy the jar name “TruckMilk<version>-Quartzjob.jar” under the path

<BONITASERVER>/server/webapps/bonita/WEB-INF/lib

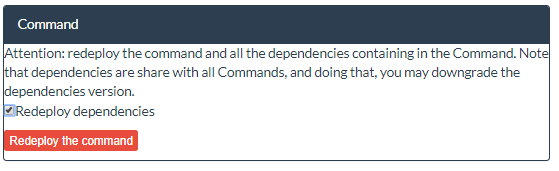
* You can copy it on your Bonita Studio: path is <STUDIO>/workspace/tomcat/server/webapps/bonita/WEB-INF/lib
* On a Cluster, you have to copy this file on each node.

Restart your server (on a studio, go to Server / Restart Web Server)

Then, access the page. Go to the tab “Maintenance” and click on START

## Force a complete installation

Dependencies are shared between command. If you need to be sure to deploy dependencies associated to the command, the Command Deployment, in the tab Maintenance, can be used



# De installation

To desinstall the page:

## Stop the Quartz job

Go to the page, tab Maintenance, and select “Stop”.

The quartz job is stopped.

## Remove the page

You can then remove the page in the Administration / resource

## Advanced setup

Nota: to remove completely the page, you can do theses action:

* Use a tool to access commands (like Command Manage, available on the community) and remove the truckmilk command.
* Remove the Quartz file

Go to

<BONITASERVER>/server/webapps/bonita/WEB-INF/lib

And remove the file “TruckMilk<version>-Quartzjob.jar”. You should need first to stop the server. On a cluster, this operation has to be done on each node.

* You can check the Quartz Table in the server to verify the jobs is correctely removed in tables qtrz\_cron\_triggers, qtrz\_triggers, qtrz\_job\_details. You can purges theses tables;

 Select \* from qrtz\_cron\_triggers where trigger\_name='trgMilktruckJob\_1';

 Select \* from qrtz\_triggers where trigger\_name = 'trgMilktruckJob\_1';

 Select \* from qrtz\_job\_details where job\_name = 'trgMilktruckJob\_1';

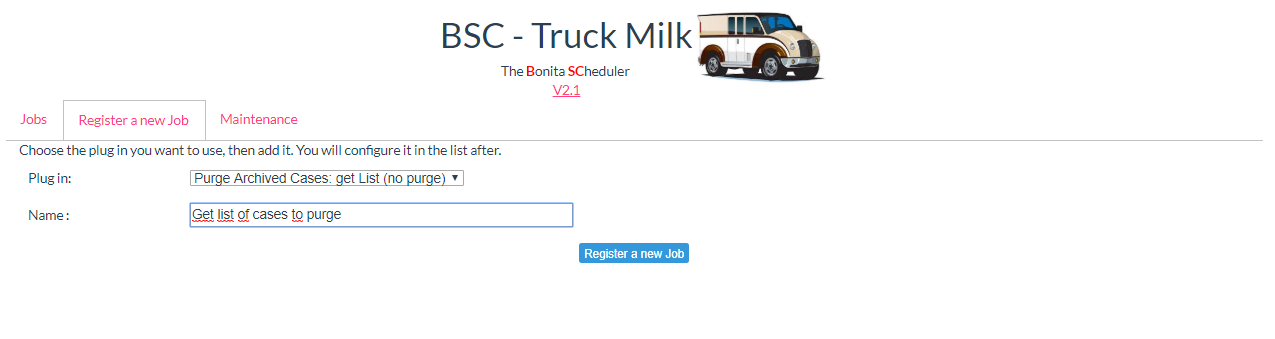
It’s not recommended to purge these table. It’s better to use the API.

# Use jobs

Truck Milk execute jobs. This chapter explain how to create a jobs, then access all parameters

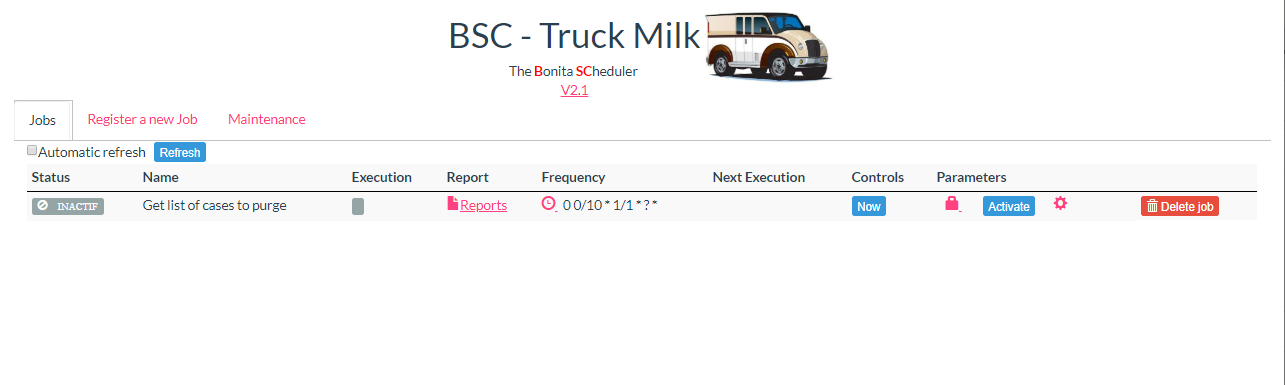
## Create a job

Click on the tab Register a new Job



Choose the Plug In in the list of existing Plug in. Give then a name. The name must be unique (it’s not possible to register two jobs with the same name).

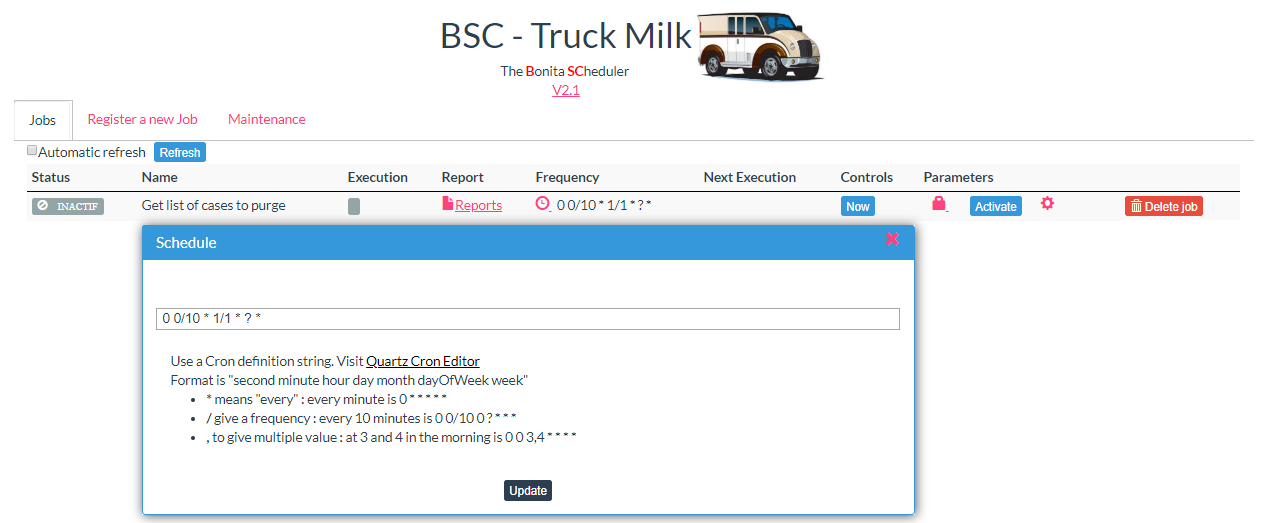
Access the Jobs tab: job is visible in the list. By default, jobs are deactivated.



## Schedule

The scheduler part specify the frequency of execution. Visit <https://www.freeformatter.com/cron-expression-generator-quartz.html> to help you to calculate the frequency.

Click on the icone  and setup the value



Click on Update to validate the change.

Note 1: the default value, 0 0/10 \* 1/1 \* ? \*, means:

At second: 00, every 10 minutes starting at minute: 00, every hour, every day starting on the 1st, every month

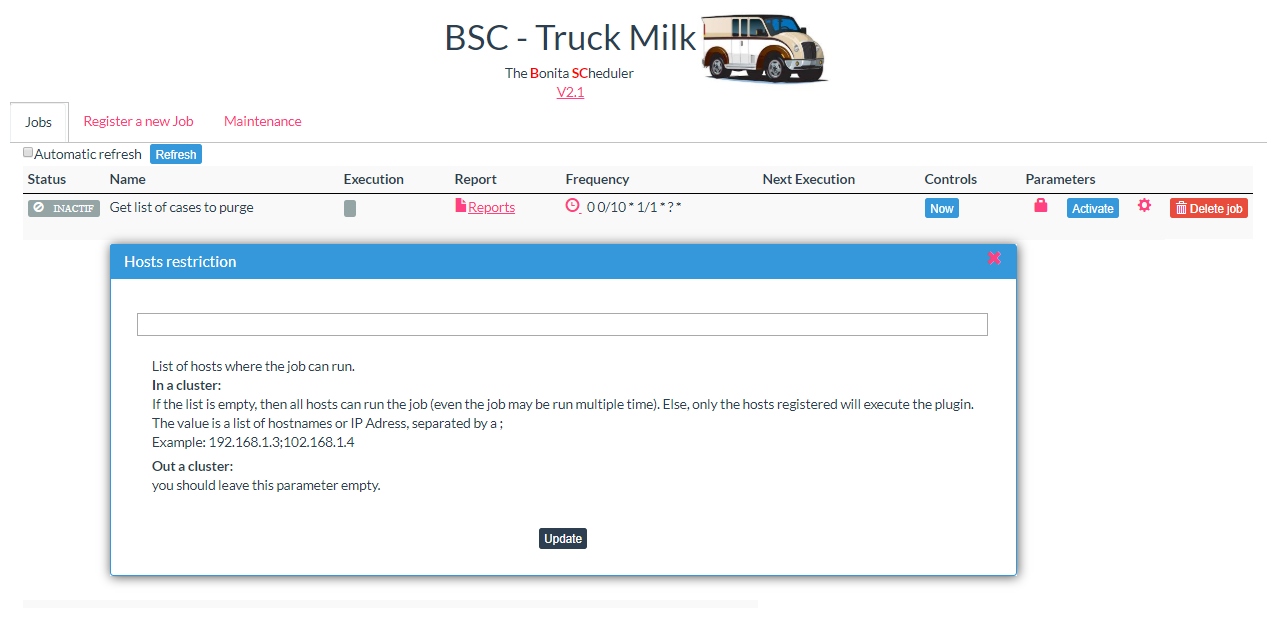
Note 2: Frequency apply only when the job is activated.

## Host restrictions

On a cluster, every nodes can execute a job. TruckMilk is based on the Bonita Scheduler, Quartz. This tools runs on each node in a cluster. When the frequency is reach, the first operation the node does is to register itself in the job, then waits two seconds, and check if the node register is still itself. This procedure avoid that two nodes starts the same job.

If you want to start the job on a particular node, and only on this one, use the Host restriction parameters, and give the IP address of the node.

Click on the icone  and setup the value



Click on Update to validate the change.

Note 1: if you have a node restriction, and this node is stopped, the job will not be executed

Note 2: you may use a list of node. Then, only one node of the list may execute the job, using the procedure given before.

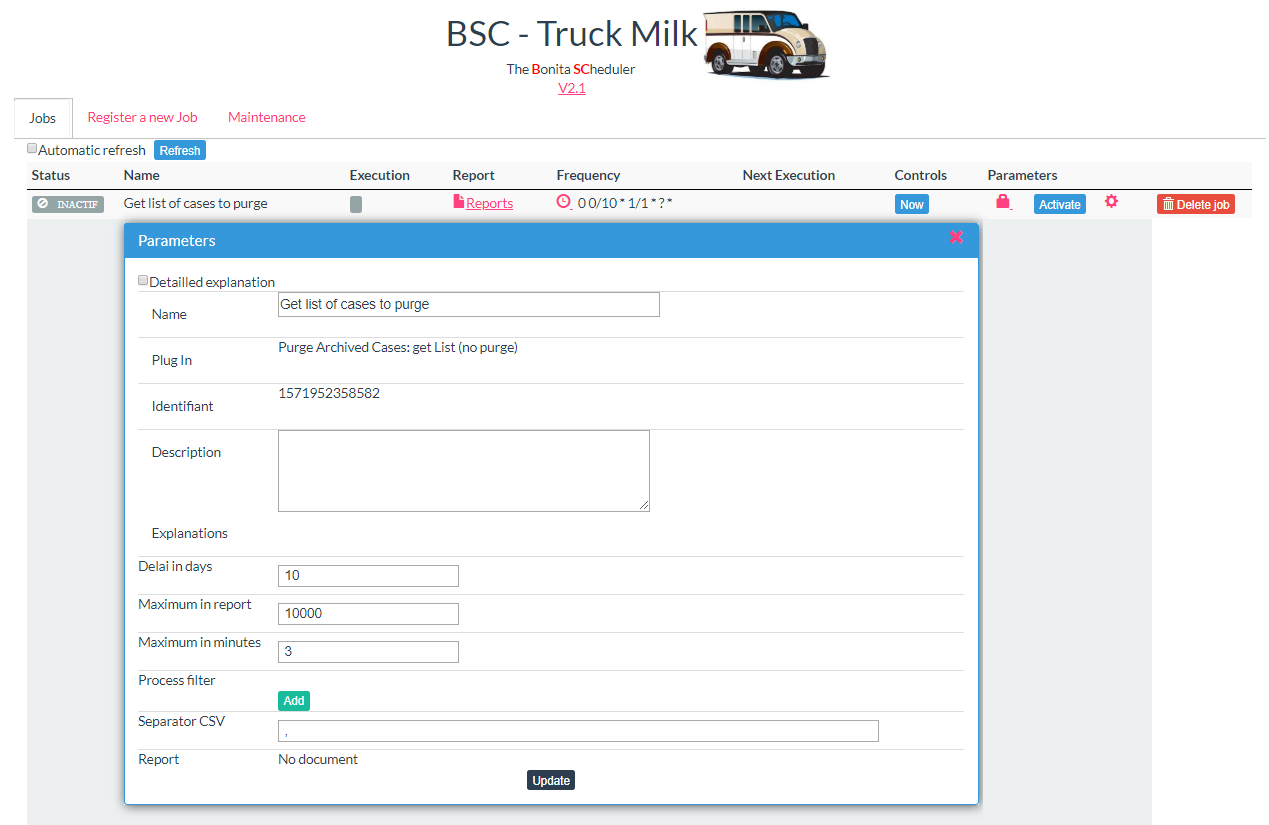
## Parameters

Jobs executing a Plug in. Each Plug in has parameters.

Each plug in has main parameters: name, description.

The identifiant is generated by Truck milk when you created the jobs and is usable on log file.

Click on  and gives the value

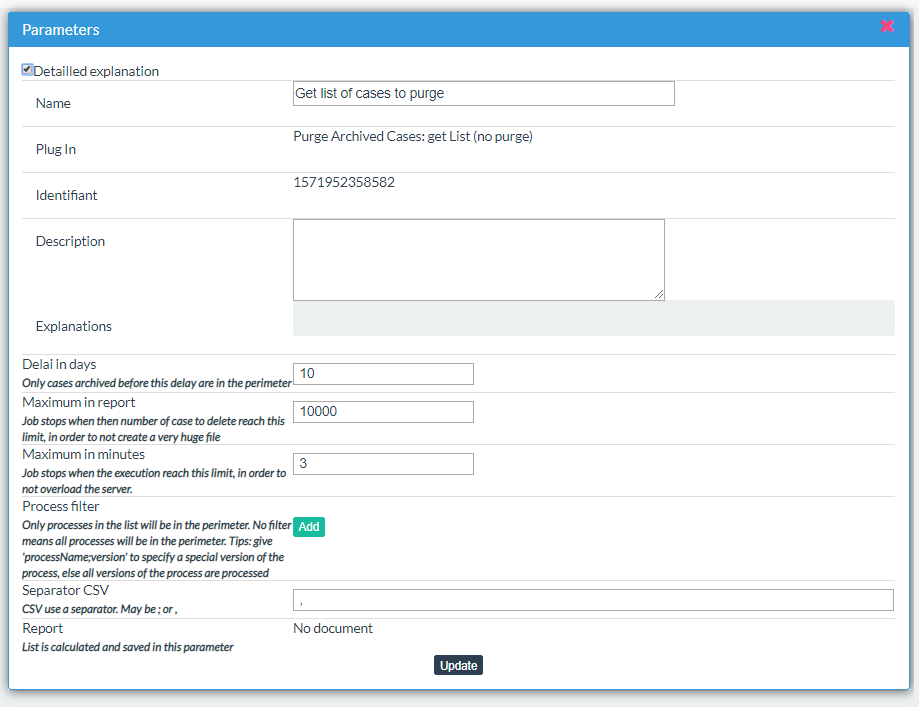


Different parameters exists:

* Integer,
* List of processes
* List of data
* Text

Click on Update to validate the change.

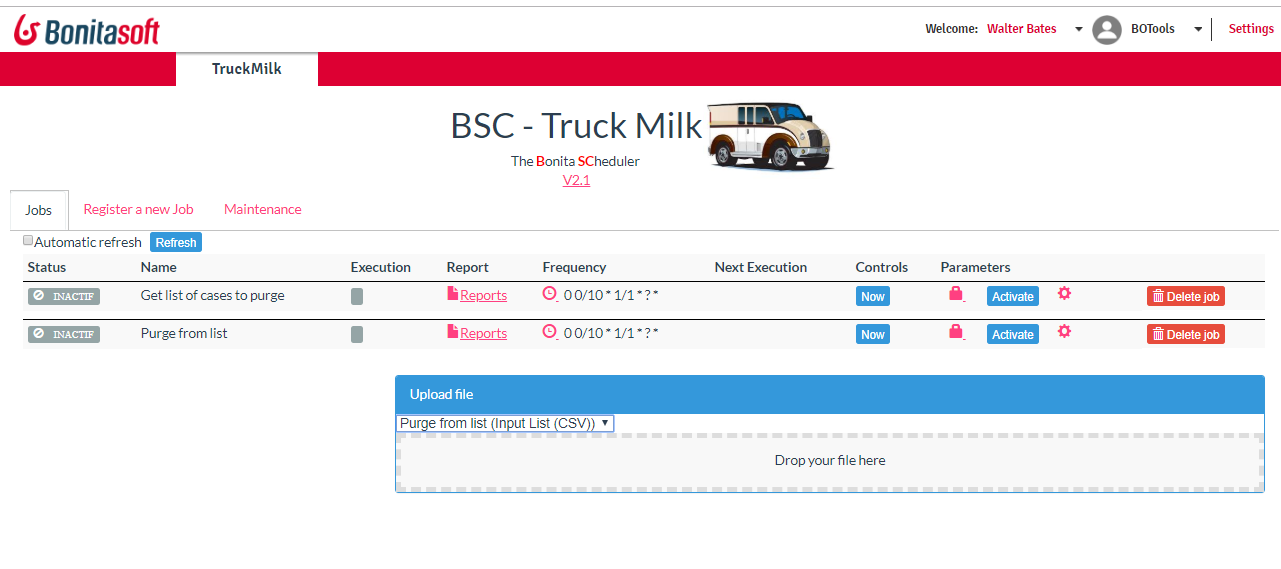
You can check the “Detailled explanation” to have more information parameter per parameter.



### Document Input parameter

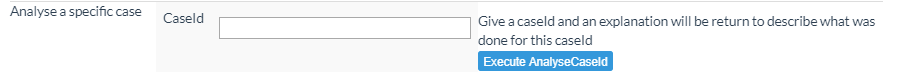
A plug in may want a document as a parameters (Plug in “Purge Archived Case: Purge from list” for example)

When a job with this kind of parameters is detected, a new box appears. Select then the job and parameter, and drag and drop the document to upload it on server.



### Test parameters

In some Plug in, you may have some tools to help you. Then, a button is visible.

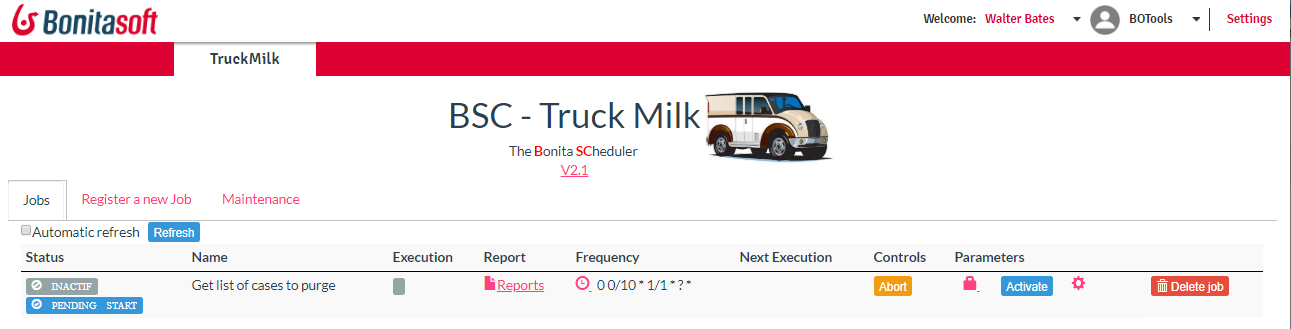


## Execute it once

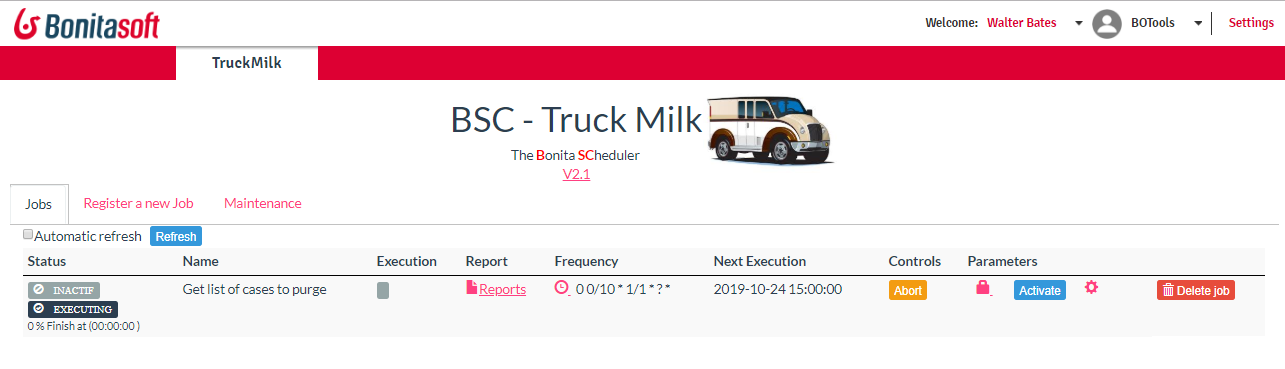
To verify parameters or have a immediate start, you can click on the button “Now”.

Jobs is then registered to be executed.

Status is moved to “Pending S.

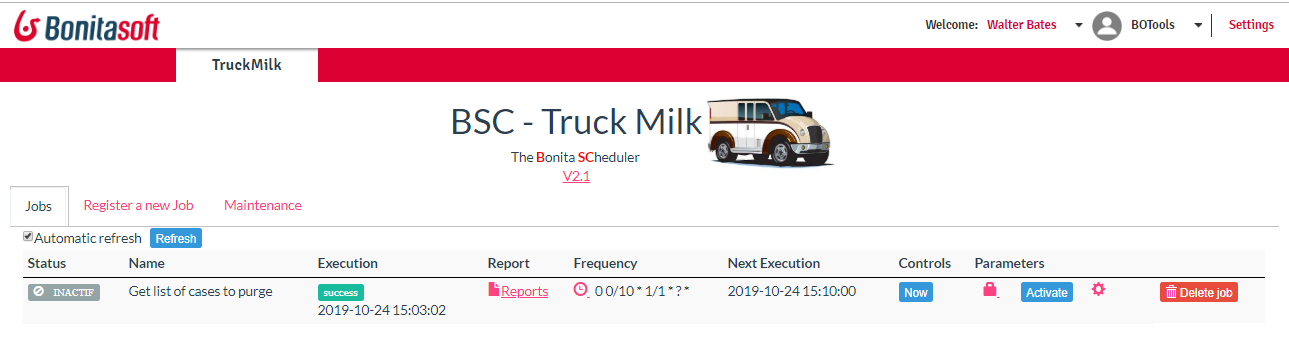


Then, job status change to “Executing”. An estimation time to finish is calculated to end when it is possible (mainly for plug in who may need time to execute).



“Abort” button may be use at any time to ask the job to finish immediately. Each Plug in test regurlarely the status, and stop during the process, at a safe point (jobs is not killed).

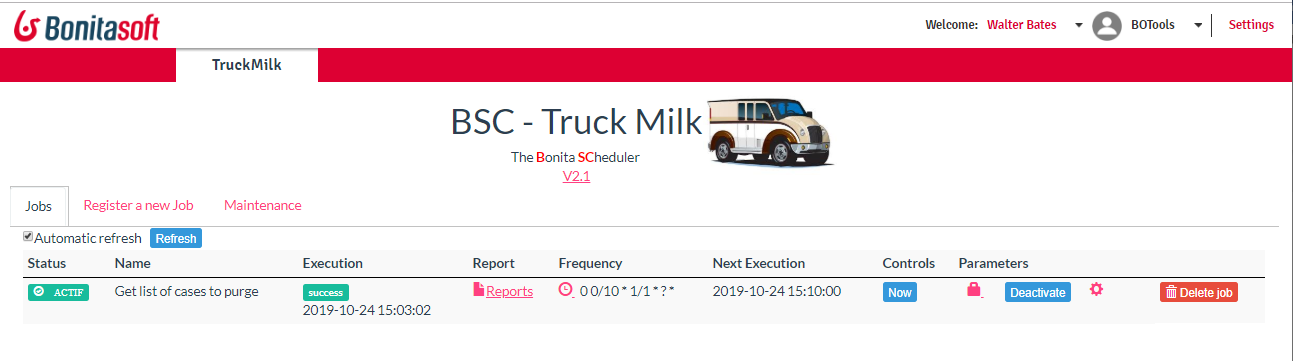
When execution finish, status change to “Success”.



Note: status may be SuccessNothing. That indicate the job runs but have nothing to do. If you asked for the list of cases to purge, and the list is empty, then the plug In return a “Sucessnothing” status. Idea is to keep then only the real execution.

## Activate / Deactivate

Activate a job by clicking on the Activate button. Truck Mil calculate the next Execution date.

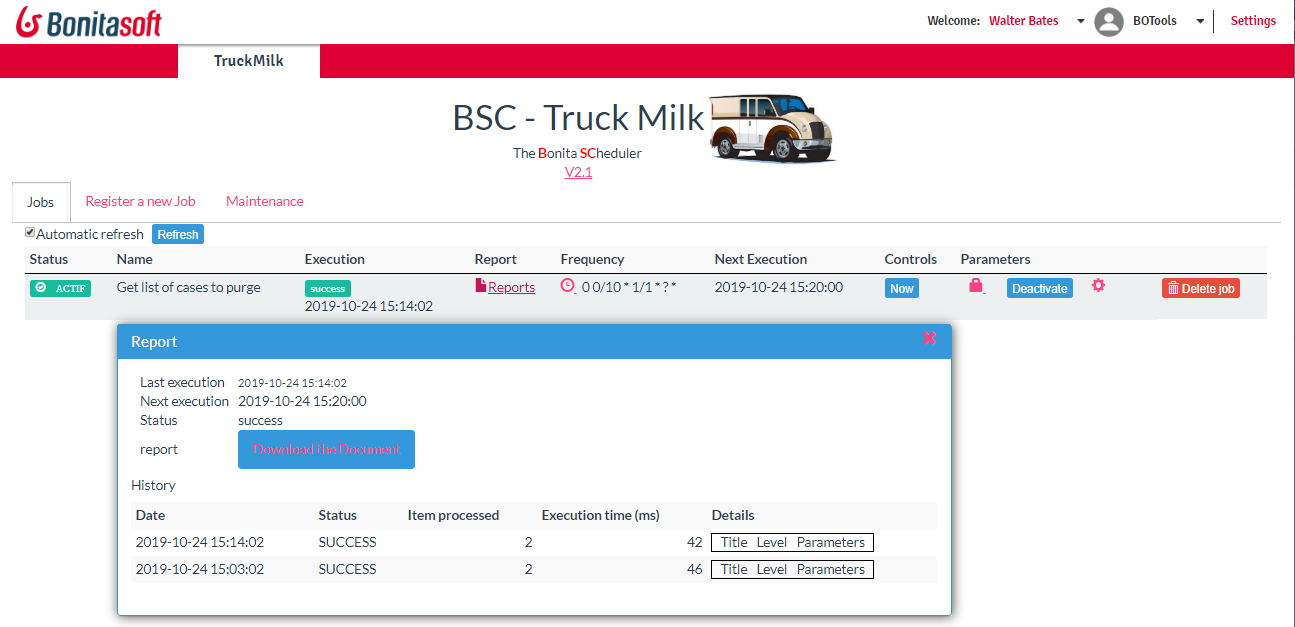


At any time, you can deactivate a job.

## Access the report

Report are accessible for each job by clicking on .

All the execution are visibles, with a status.



Note 1: when the Plug in produce a document, you can download it. Only the last generated document is saved.

Note 2: the history keep only the Success and Error execution. If the Plug in return a SuccessNothing, it is not saved in the list, to keep history with real information.

Note 3: Only the last ten executions are kept.

# Embeded plug in

## Delete case

The Plug in delete cases, archived or not. A list of processes can be set as parameters, else all processes. Use this plug in when its necessary to purge a lot of cases.

Deletion of the case may not works due to a transaction timeout: to delete a process, Bonita Engine open a transaction, then delete all cases. When there are too much case to remove, transaction may failed. Secondly, this operation may need a lot of time to be finished.

The parameter “Maximum cases deletion” is use to specify the maximum number of case to delete. When the number is reach, the job finish, it will re-start at the next execution, so cases may be removed “page per page” to avoid any overloading on the server.

## Email User tasks

This Plug in calculated, for a user, all visible tasks, i.e., all tasks visibles in the portal, in “my tasks”.

Then, one (and only one) email is send to the user, with the list of all tasks.

Plug in required a User Profile as parameters: all users registered in the profile will receive an email if they have active tasks.

The content of email may be defined, using place holder:

* Assignedtask list all tasks assigned to the user
* Pendingtasks contains all tasks assigned (except the assigned tasks)

For example, the content may be

Your assigned task <br>{{assignedtasks}}<p>Your pending tasks:<br>{{pendingtasks}}

Nota: this plug in need the Email JAR file installed. Check tab Maintenance / Information to verify that the JAR is correctly installed.

## Ping

Ping Plug In is here for test usage

## Purge Archived Case

This Plug in purge archived case. A delay may be set in parameter, to purge case with three months delay for example.

## Purge Archived Case : Get List

This Plug In calculated a list of cases to purge, but not purge them. The list is a CSV file, containing the caseid, the process name. A status column is ready to be fulfilled.

A delay can be set in parameter, and a list of processes.

Example of CSV:

caseid;processname;processversion;archiveddate;status

5149;ExpenseNote;1.0;18/10/2019 15:02;

5150; ExpenseNote;1.0;18/10/2019 15:02;

5151;VacationRequest;1.0;18/10/2019 15:02;

## Purge Archived Case: Purge From List

This Plug In works with the previous plug In. it accepts the same list, and checl the status of each line. If the status is DELETE, then the case is removed.

Example of CSV to upload:

caseid;processname;processversion;archiveddate;status

5149;ExpenseNote;1.0;18/10/2019 15:02;DELETE

5150; ExpenseNote;1.0;18/10/2019 15:02;

5151;VacationRequest;1.0;18/10/2019 15:02;DELETE

## Replay Failed Tasks

Replay Fail task re-execute all tasks in a Failed mode. The number of retry, and the delay between two tentative are parameters.

Note: the Bonita Engine must accept to replay a failed task. Only the Entreprise or Performance subscription allow that.

## SLA

Tasks must have some due date to be executed (example, execute the task in 10 days). Before the delay is reach, different actions may be necessary:

* Send a reminder after 6 days (i.e. at 60% of the delay) to all users who can execute the task
* Send a second reminder after 9 days (90%)
* Send a alert reminder if the delay is over than 1 days (110%)
* Assign the task to a different person if the delay is over 13 days (130%)

A design in the process is possible to reach theses operations, based on “non interruptible boundary event”. When process have a lot of human tasks, and rules are important, the design may become very complexe. Secondly, if a rule change (add a new trigger, change the trigget at 90 to 80%), the process must be redeployed.

This Plug In does theses operations, and can be update at any time.

The SLA operation is based:



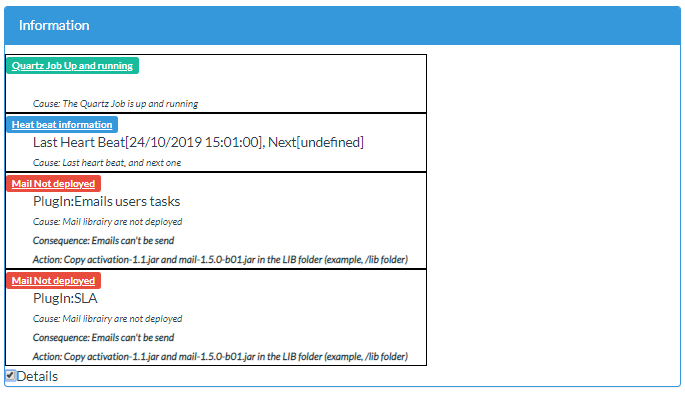
* A task name. If this parameter is empty, the rule is applied on all tasks
* A Percent Threashold. This parameters is active only if the task has a Due Date. Then, the percentage is calculated between the Start Date and the Due Date. When the due date is reach, the percent is 100%. It’s possible then to apply a rule over 100% (due date expired)
* Action: multiple action are possible. Click on detailExplanation to have a list of explanation. Main actions are EMAILUSER, EMAILACTOR, EMAILCANDIDATE,ASSIGNUSER, ASSIGNSUPERVISOR

# Maintenance

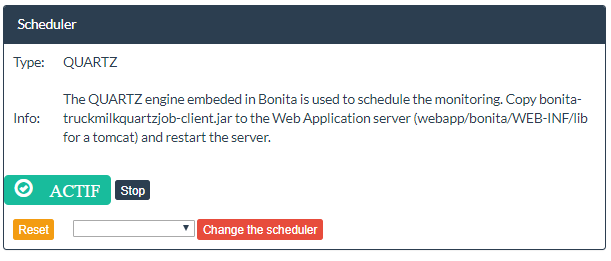
The Maintenance tab group some operations:

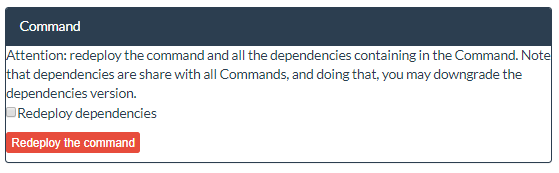
## Information

This panel contains main information on the Scheduler.









# Build a new plug in

This part explains how to develop a new Plug In. Don’t hesitate to create and send to the community any new plug in. This plug in will be available then on any new release of the page and is maintained. Please note the plug in must be general (don’t hard code any process name, any information, use parameters).

## Clone the repository

Clone the github repository

<https://github.com/Bonitasoft-Community/page_truckmilk>

You can import the process under Eclipse.

## Compile, deploy locally

To compile the page, just execute

mvn install

This order

* Download all needed component from Maven
* Compile the JAVA and generate a JAVA library
* Generate the page resource (a ZIP file, containing all libraries, HTML, resources)
* Deploy the page locally on your server, using localhost:8080 and walter.Bates to realize the job (if you need to deploy on a different server, with a different user name, modify the pom.xml)
* Create a profile named BOTools, and reference the page in this profile

To test the page, click again on the page in the menu bar: Bonita reload the page then.

## Create a new plug in

A Plug In is a JAVA classes.

By convention, create the class under org.bonitasoft.truckmilk.plugin. Start it by “Milk…. This class should extend the class MilkPlugIn.

Then reference the new class in the MilkPlugInFactory, method collectListPlugIn() : this is the condition for the new plug in to appears in the list of “new job to register”.

Look the Plug In “ping” (MilkPing.java). This is an example.

In the plug in, you should define:

* getDefinitionDescription() : you define the description, name, description, plus the list of parameters
* execute() : the core of the plug in
* checkPlugInEnvironment() : you can use any detection in this method to verify the environment. Do you have access to all Java Library?
* checkJobEnvironment(): here the check according the job. You have access of all parameters of the job. For example, the parameters give a Data source value. Can you connect to this data base?

## Using parameters

## Advancement

## State of the art

* Don’t hard code any value in the plug in. Use parameters. For example, if your plug-in have a delay to calculate the scope (delay when a process was deployed, delay when a case was archived), don’t hard code this delay, ask it as a parameters.
* Use parameters. Your plug in works on process? Add a parameter to filter the scope of processes. If the parameter is empty, then check all processes
* Use the correct parameters type: for a process, use ARRAYPROCESS, not STRING.
* If your plug in is very heavy (for example, purge cases can need time if you want to purge 100 000 cases), then use the setAvancement() method. Administrator will see the advancement and can stop it.
* For a long running, ask as a parameter a maximum operation, or a maximum time to process. Then, you will not execute a “4 days works”, and the administrator can configure the treatment to run only 3 hours every night for example.
* Think big. Keep in mind your plug in can work on a large panel of input, so you have to create a robust plug in. use the setAdvancement(), allow the work to be stop after a certain number of items.
* Use the BEvent library. To report information, error, prefer the BEvent library. A BEvent contains the error title, plus the consequence and action to fix it. Who is the best person to document the “what to do”? The developer when he references the error. It’s a (little) more works for the developer, but really help the administrator. Then, because each BEvent has a unique number, it’s easy to find in the code where is the error.

HEADQUARTERS

PARIS, FRANCE

76 boulevard de la République

92100 Boulogne-Billancourt

EMEA, ASIA & LATIN AMERICA

GRENOBLE, FRANCE

32, rue Gustave Eiffel  
38000 Grenoble

NORTH AMERICA

SAN FRANCISCO, USA

44 Tehama Street  
San Francisco, CA 94105

NEW YORK, USA

33 Nassau Avenue

Brooklyn NY 11222