Truckmilk Manual

Jobs executor

June 2019

Sommaire

[1. Introduction 4](#_Toc52447231)

[2. Usage 4](#_Toc52447232)

[3. Installation 4](#_Toc52447233)

[3.1. Get the resource 4](#_Toc52447234)

[3.2. Install the page 4](#_Toc52447235)

[3.3. Install the Quartz 4](#_Toc52447236)

[3.4. Force a complete installation 5](#_Toc52447237)

[4. De installation 6](#_Toc52447238)

[4.1. Stop the Quartz job 6](#_Toc52447239)

[4.2. Remove the page 6](#_Toc52447240)

[4.3. Advanced setup 6](#_Toc52447241)

[5. Use jobs 6](#_Toc52447242)

[5.1. Create a job 7](#_Toc52447243)

[5.2. Schedule 7](#_Toc52447244)

[5.3. Parameters 8](#_Toc52447245)

[5.3.1. Document Input parameter 9](#_Toc52447246)

[5.3.2. Test parameters 10](#_Toc52447247)

[5.4. Execute it once 10](#_Toc52447248)

[5.5. Activate / Deactivate 11](#_Toc52447249)

[5.6. Access the report 11](#_Toc52447250)

[6. Embedded plug in 12](#_Toc52447251)

[6.1. Delete case 12](#_Toc52447252)

[6.2. Email User tasks 12](#_Toc52447253)

[6.3. Ping 13](#_Toc52447254)

[6.4. Purge Archived Case 13](#_Toc52447255)

[6.5. Purge Archived Case: Get List 13](#_Toc52447256)

[6.6. Purge Archived Case: Purge from List 13](#_Toc52447257)

[6.7. Replay Failed Tasks 13](#_Toc52447258)

[6.8. SLA 14](#_Toc52447259)

[7. Maintenance 14](#_Toc52447260)

[7.1. Information 14](#_Toc52447261)

[8. Build a new Plug-in 15](#_Toc52447262)

[8.2. Clone the repository 15](#_Toc52447263)

[8.3. Compile, deploy locally 16](#_Toc52447264)

[8.4. Create a new Plug-in 16](#_Toc52447265)

[8.5. Use Bonita Events 17](#_Toc52447266)

[8.6. Check Plug-in environment 17](#_Toc52447267)

[8.7. Check Job environment 17](#_Toc52447268)

[8.8. Definition description 18](#_Toc52447269)

[8.9. Using parameters 18](#_Toc52447270)

[8.9.3. Basic type 19](#_Toc52447271)

[8.9.4. Complex type 19](#_Toc52447272)

[8.9.5. Documents type 20](#_Toc52447273)

[8.9.6. Bonita type 20](#_Toc52447274)

[8.9.7. Special usages 21](#_Toc52447275)

[8.10. Conditions with parameters 21](#_Toc52447276)

[8.11. Execution 22](#_Toc52447277)

[8.12. Measures 23](#_Toc52447278)

[8.13. Chronometers 23](#_Toc52447279)

[8.14. Advancement 23](#_Toc52447280)

[8.15. Stop mechanism 24](#_Toc52447281)

[8.16. Document management 24](#_Toc52447282)

[8.17. State of the art 25](#_Toc52447283)

Content

# Introduction

This document explains how to use the TruckMilk Page.

# Usage

The page may be use in different context:

# Installation

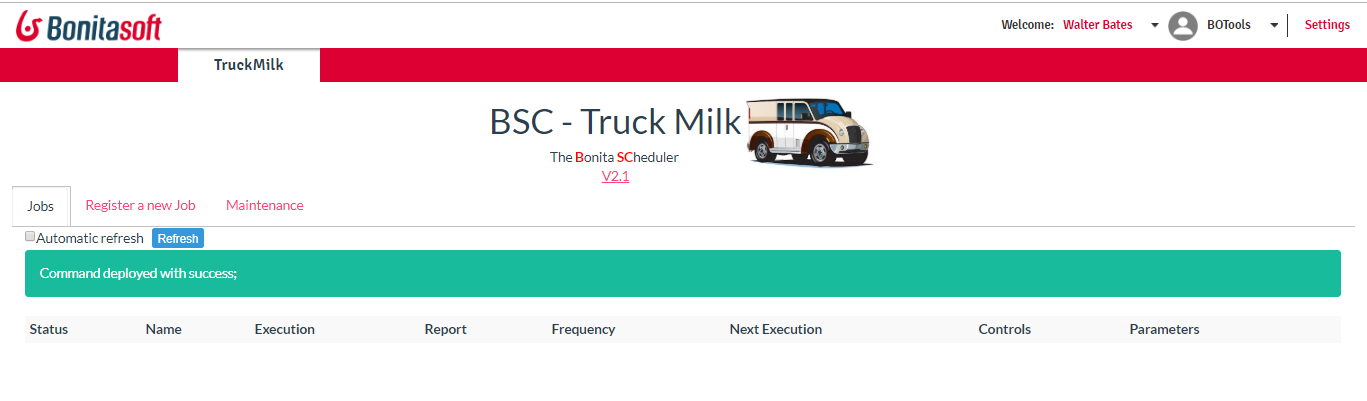
## Get the resource

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

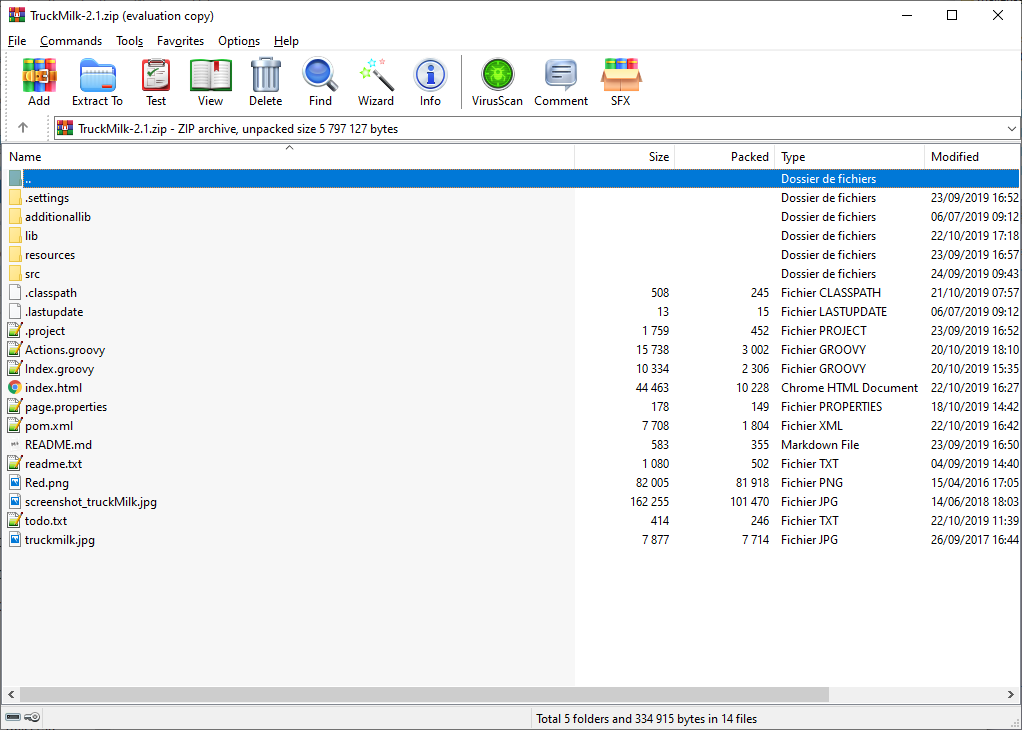
A Bonita Command is a JAVA Class deployed on the server. This allow the controller to check and start new jobs even if user is disconnected.

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

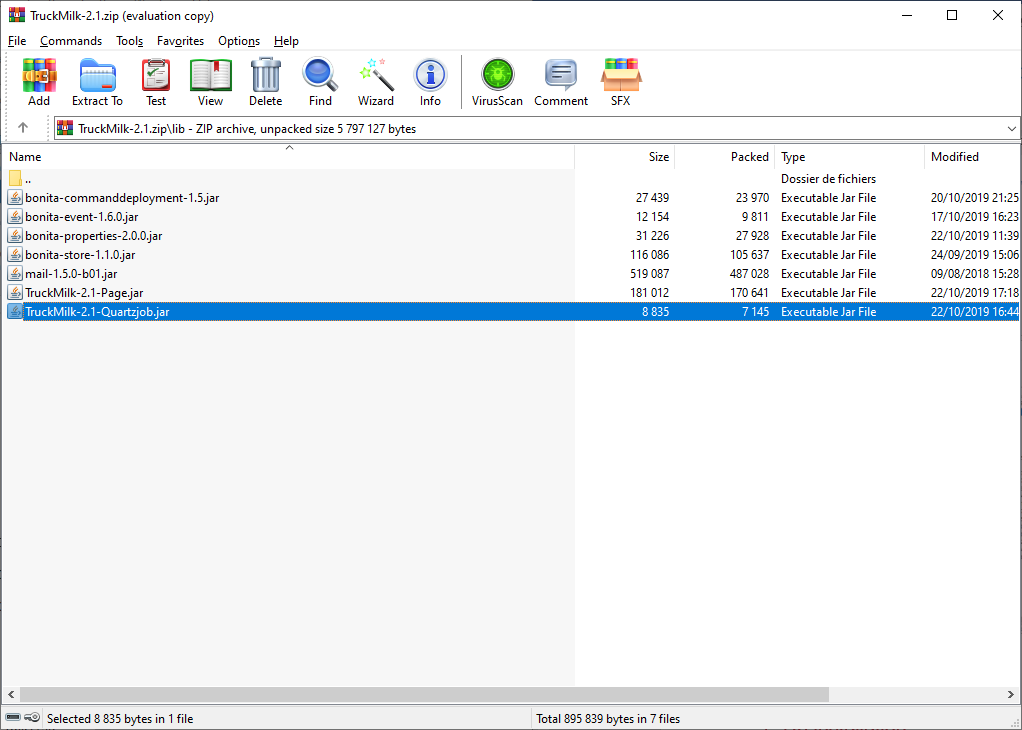
The second step consist to install the Quartz Job JAR file.

## 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

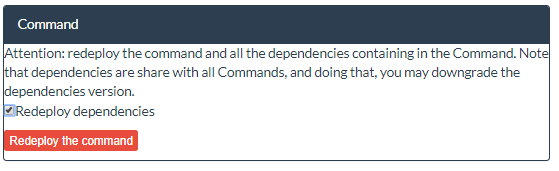
* You can copy it on your Bonita Studio: path is <STUDIO>/workspace/tomcat/server/webapps/bonita/WEB-INF/lib
* On a Cluster, you have to copy this file on each node.

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, On the maintenance tab, use the undeploy button. This function

* Remove the TruckMilk command,
* Remove all Quartz job

To finish, 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 correctly removed in tables qtrz\_cron\_triggers, qtrz\_triggers, qtrz\_job\_details;

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

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

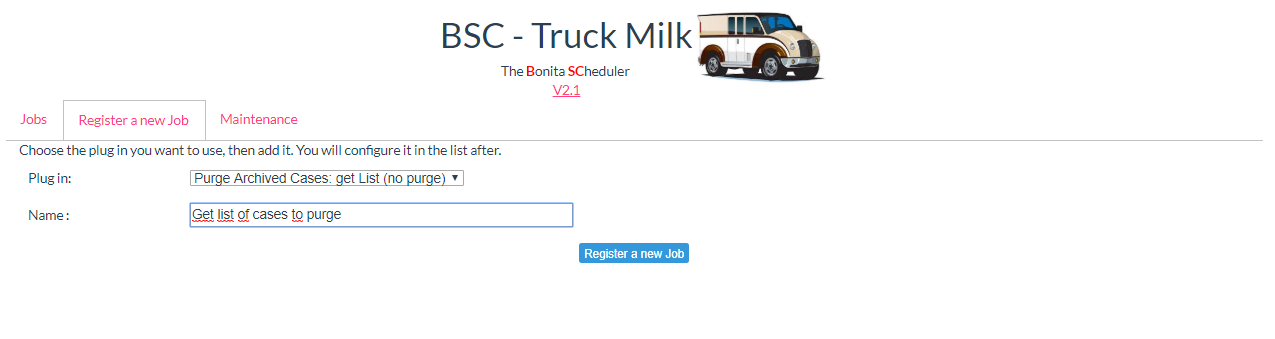
 Select \* from qrtz\_job\_details where job\_name = 'trgMilktruckJob\_1';

# Use jobs

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

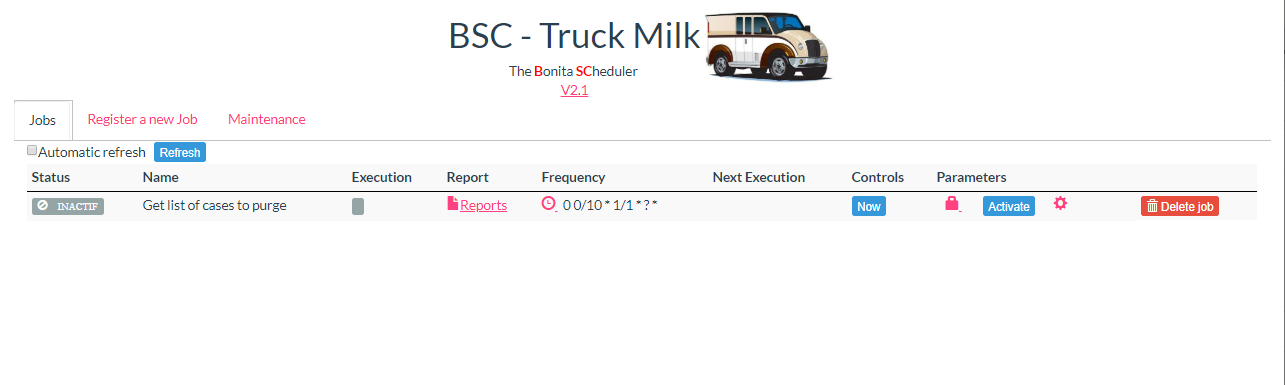
## Create a job

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).

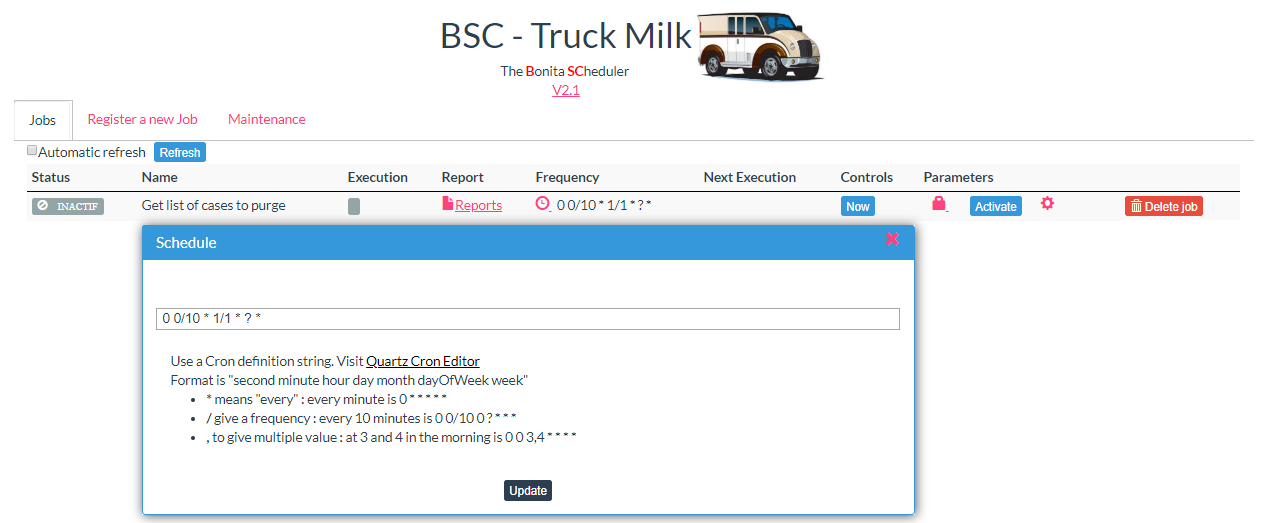
Access the Jobs tab: job is visible in the list. By default, jobs are deactivated.



## 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.

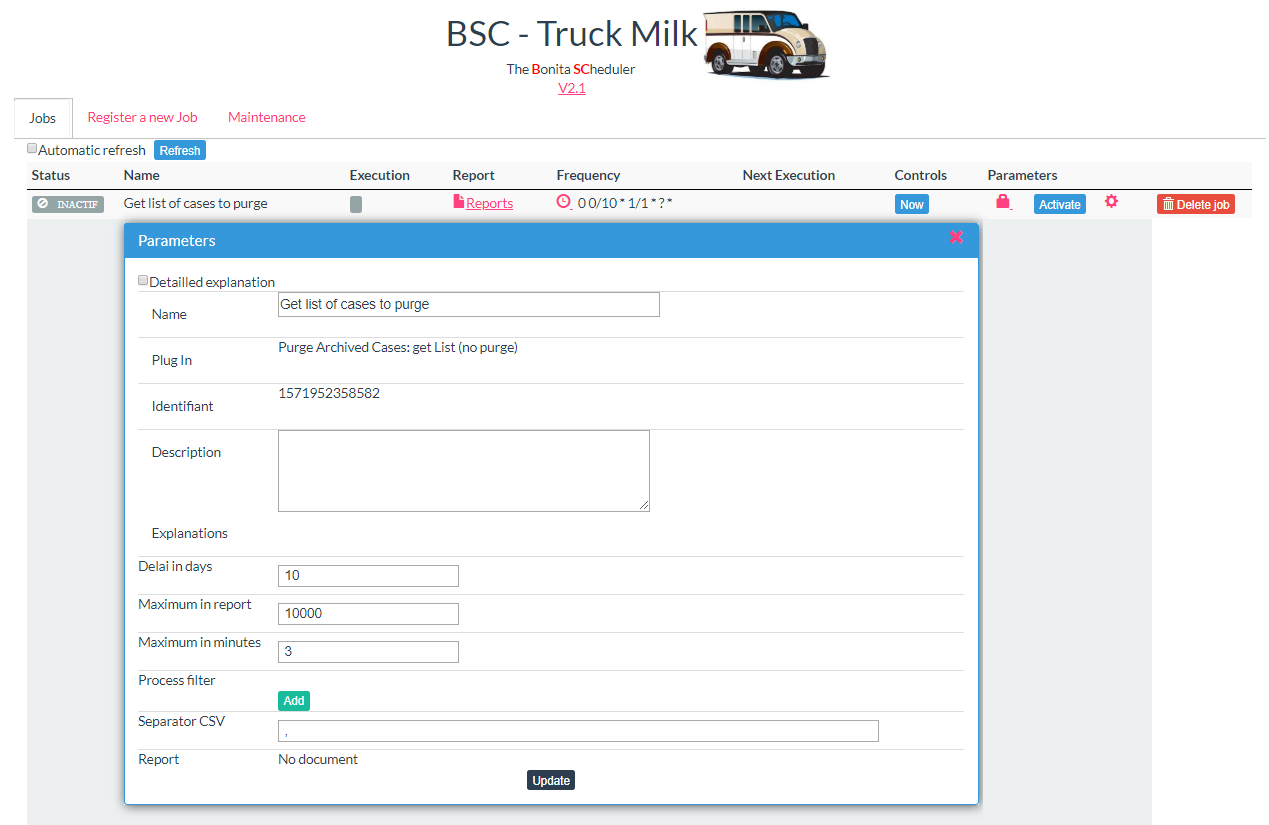
## Parameters

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

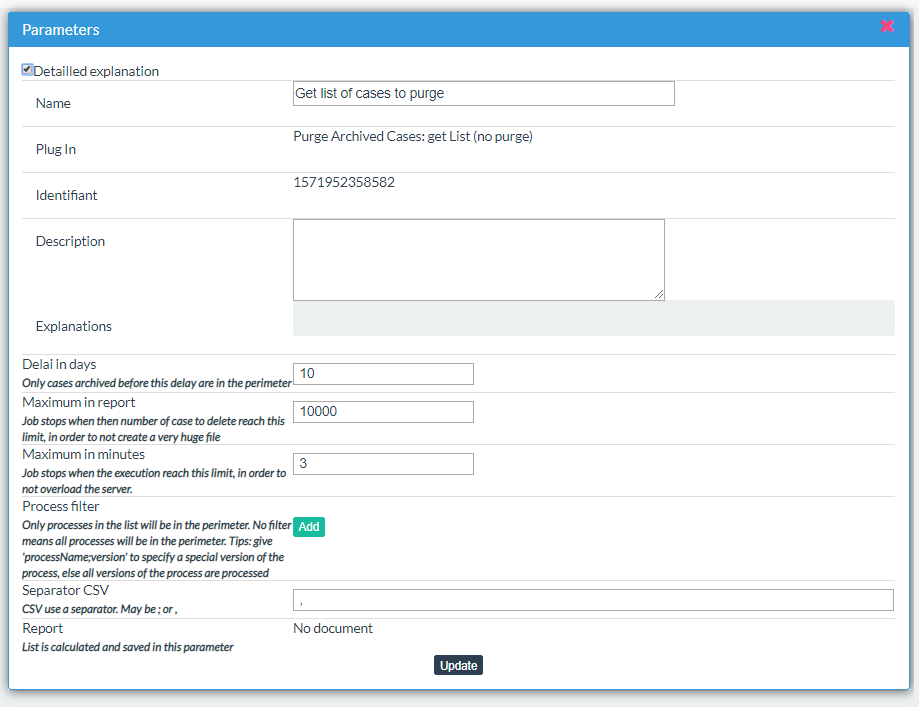


Different parameters exist:

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

Click on Update to validate the change.

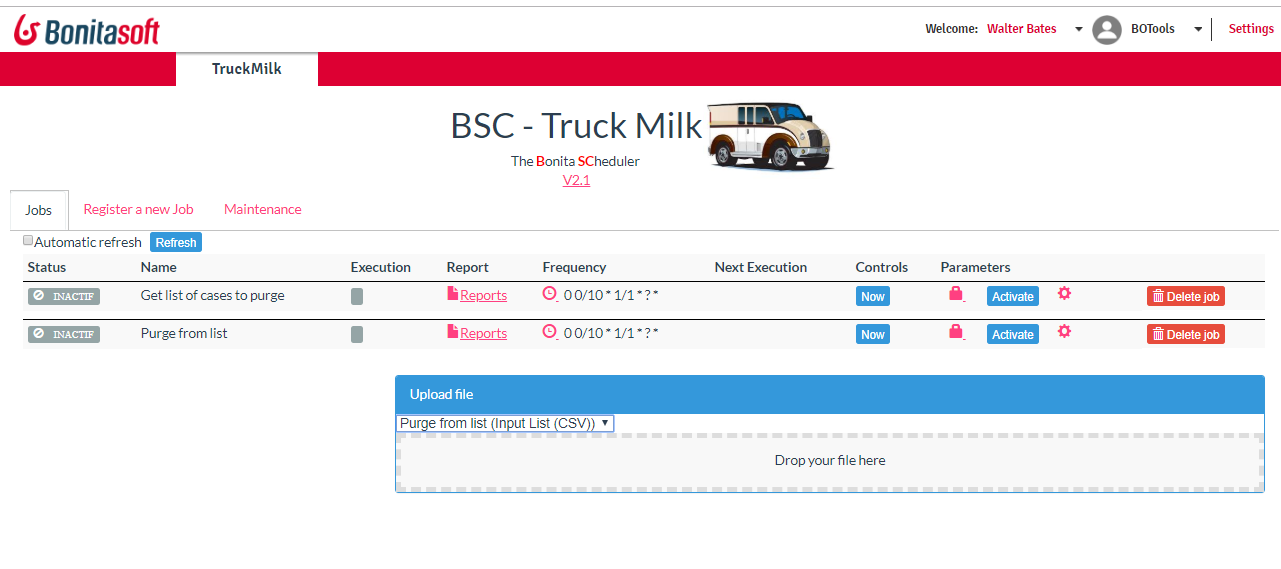
You can check the “Detailed explanation” to have more information parameter per parameter.



### Document Input parameter

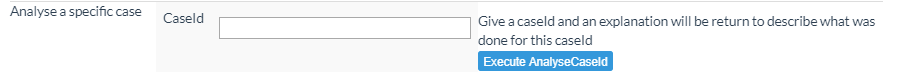
A plug in may want a document as a parameter (Plug in “Purge Archived Case: Purge from list” for example)

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.

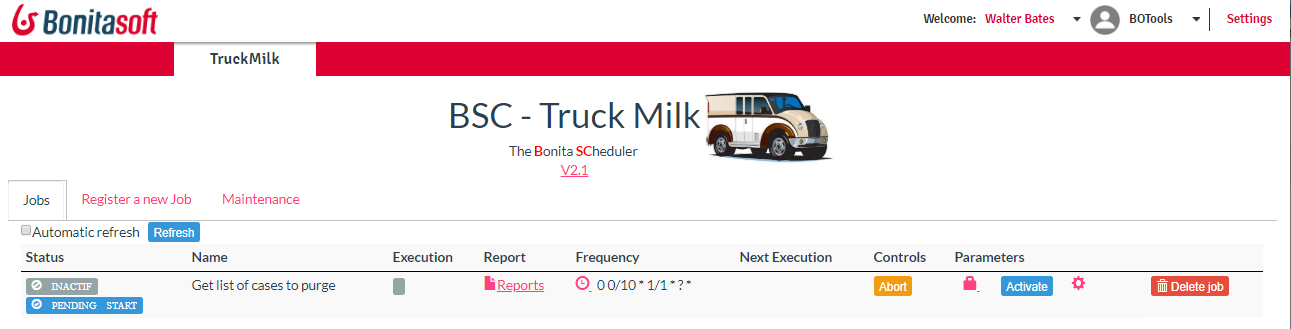


## Execute it once

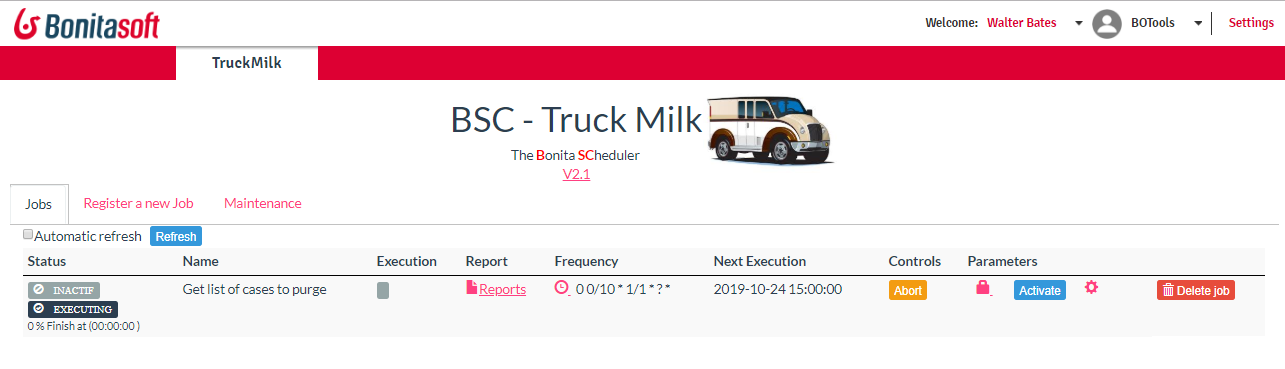
To verify parameters or have an immediate start, you can click on the button “Now”.

Jobs is then registered to be executed.

Status is moved to “Pending Start”.

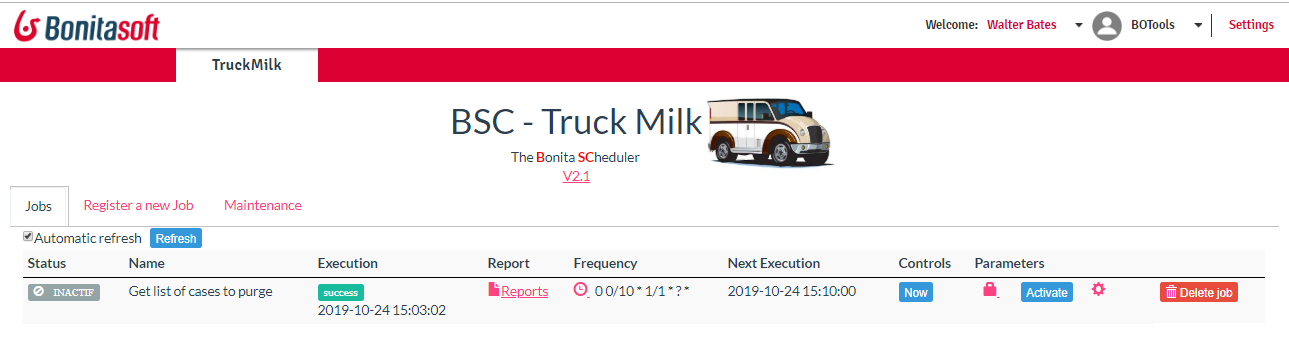


Then, job status change to “Executing”. An estimation time to finish is calculated to end when it is possible (mainly for plug in who may need time to execute).



“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).

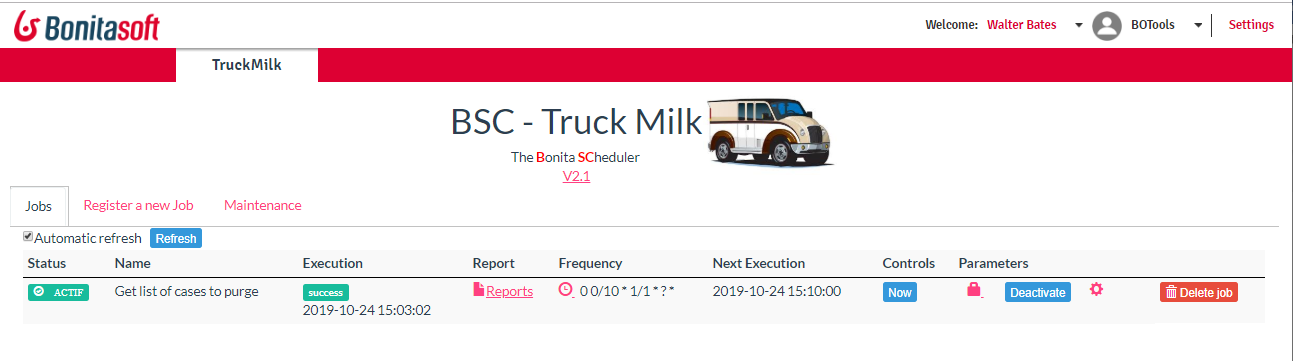
When execution finish, status change to “Success”.



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.

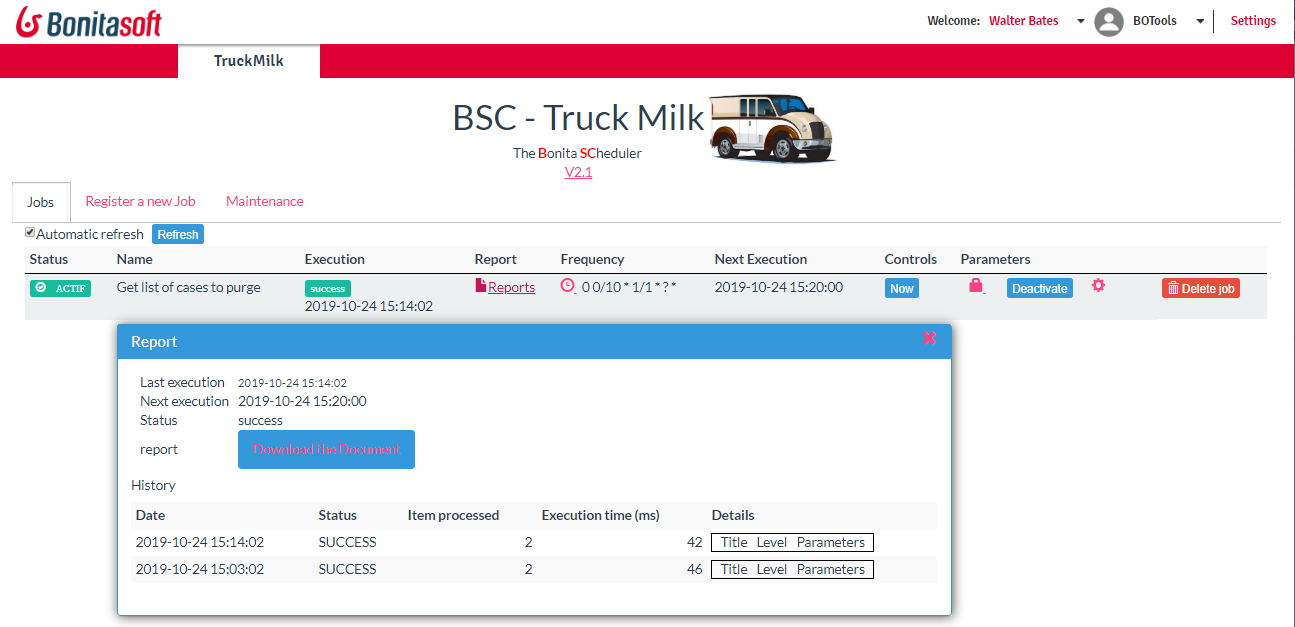


At any time, you can deactivate a job.

## Access the report

Report are accessible for each job by clicking on .

All executions are visible, 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 keeps 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.

# Embedded plug in

## Cancel/Delete Active Cases (active)

|  |  |
| --- | --- |
| Name | Cancel/Delete Active Cases (active) |
| Category | CASES |
| Job stopper | BOTH |

Cancel(or delete) all cases older than a delay, or inactive since a delay

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Filter on process | Job manage only process which mach the filter. If no filter is given, all processes are inspected |
| Delai | Only cases archived before this delay are in the perimeter |
| operation: Build a list of cases to operate, do directly the operation, or do the operation from a list | Result may be the cancellation or deletion , or build a list, or used the uploaded list |
| Trigger to detect case to work on | Case Start Date : the cases started before the delay are in the perimeter, else Case Last Update Date only cases without any operations after the delay |
| Action on cases : cancellation or deletion | Cases are cancelled or deleted |
| Separator CSV | CSV use a separator. May be ; or , |
| Report Execution | List of cases managed is calculated and saved in this parameter |
| Cases to delete/cancel | List is a CSV containing caseid column and status column. When the status is 'DELETE' or 'CANCELLED' or 'OPERATE', then the case is operate according the status (if OPERATE, then the job operation is used) Example: caseId;status 342;DELETE 345;DELETE |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Time Execution | Time to execute the plug in (in ms) |
| Nb items processed | Number of items the job processed during the execution |

## Clean Archived Dross

|  |  |
| --- | --- |
| Name | Clean Archived Dross |
| Category | CASES |
| Job stopper | BOTH |

Check dross in archived tables

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Operation | Detect or clean all Drosses in archived table Only detection: Only detect. Purge: Purge records. |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Time Execution | Time to execute the plug in (in ms) |
| Nb items processed | Number of items the job processed during the execution |

## Delete Cases (active and archived)

|  |  |
| --- | --- |
| Name | Delete Cases (active and archived) |
| Category | CASES |
| Job stopper | BOTH |

Delete all cases in the given process, by a limitation given in parameters

### Documentation

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 work 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 fail. Secondly, this operation may need a lot of time to be finished.

The parameter ″Maximum cases deletion″ is used 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.

Attention, all deletion is definitive

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Filter on process | Job manage only process which mach the filter. If no filter is given, all processes are inspected |
| Delay to delete case, based on the Create date | All cases STARTED before this delay will be purged, even if they have a task execution after |
| Measure |  |
| Name | Explanation |
| Time Execution | Time to execute the plug in (in ms) |
| Nb items processed | Number of items the job processed during the execution |

## Delete Duplicate Tasks

|  |  |
| --- | --- |
| Name | Delete Duplicate Tasks |
| Category | TASKS |
| Job stopper | BOTH |

3 operations: PURGE/ GET LIST / PURGE FROM LIST. Purge (or get the list of) archived case according the filter. Filter based on different process, and purge cases older than the delai. At each round with Purge / Purge From list, a maximum case are deleted. If the maximum is over than 100000, it's reduce to this limit.

PlugIn does a executeQuery(update flownode\_instance set STATECATEGORY='CANCELLING'

and then it does a processAPI.setActivityStateByName( ActivityStates.CANCELLED\_STATE)

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| operation | Detect the list of duplicate task or delete them |
| Process Filter | Give a list of process name. Name must be exact, no version is given (all versions will be purged) |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Task deletion error | Number of deletion error |
| Time Execution | Time to execute the plug in (in ms) |
| Nb items processed | Number of items the job processed during the execution |
| Tasks deleted | Number of task deleted |

## Delete Processes (ENABLED or DISABLED)

|  |  |
| --- | --- |
| Name | Delete Processes (ENABLED or DISABLED) |
| Category | PROCESSES |
| Job stopper | BOTH |

Delete processes. If process has Active or Archived cases, they may be purged too

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Scope | Scope of deletion |
| Operation | Detect or delete processes Only detection: Only detect. Deletion: Purge records. |
| Process in the perimeter | Job manage only process which mach the filter. If no filter is given, all processes are inspected |
| Exclusion list | This processes are not in the scope of the job |
| Separator CSV | CSV use a separator. May be ; or , |
| List of Processes | List of processes is is calculated and saved in this parameter |
| Disabled | Deletion based on processes DISABLED |
| Disabled processes | Delete all Diseable processes |
| Purge Cases | Purge cases policy. If the process is called as a sub process, sub case is not purged |
| No activity in this period | Process is deleted only if there is not operation in this delay (based on the case STARTED or Last Update Modification - an task executed after is not take into account) |
| Keep Last Version of the process | Keep the last version of the process |
| Unused | Deletion based on processes not used |
| Process not used | Delete all Process not used in the delay (No Active/Archived case STARTED in the period) |
| Purge Cases | Purge cases policy. If the process is called as a sub process, sub case is not purged |
| No activity in this period | Process is deleted only if there is not operation in this delay (based on the case STARTED or Last Update Modification - an task executed after is not take into account) |
| Keep Last Version of the process | Keep the last version of the process |
| Maximum Version | Deletion based on process version |
| Max version | Keep a number of version for a process. Older version than the max are deleted. |
| Disable after version number | Keep a number of version for a process. Older version than the max are disabled. Example, set to 3, you keep only the 3 first enable, process, after it will be disabled |
| Delete after version number | All processes under this number for a process are deleted (based on the deployment date). Example, set to 5, you keep only the 5 first process, after it will be deleted |
| Purge Cases | Purge case policy. If the process contains cases, it will be purged according the policy |
| Scope | Delete process policy. With a ONLYDISABLED process, before deleting a process, check if Only Disabled process can be deleted, or Enable process as well |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Time Execution | Time to execute the plug in (in ms) |
| Nb items processed | Number of items the job processed during the execution |

## Emails users’ tasks

|  |  |
| --- | --- |
| Name | Emails users tasks |
| Category | TASKS |
| Job stopper | BOTH |

For all users’ part of the given profile(s), an email is send with a link on all pending tasks

### Documentation

This Plug in calculated, for a user, all visible tasks, i.e., all tasks visible 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}}

Note: this plugin need the Email JAR file installed. Check tab Maintenance / Information to verify that the JAR is correctly installed.

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Profiles User | Give a list of Profiles User. All users referenced in theses profiles are inspected, to detet if they have tasks to execute |
| SMTP Parameters | To send an email, SMTP parameters is required to connect a SMTP server |
| SMTP Parameters | Smtp parameters: host:port:username:password |
| Bonita Host | In the Email, the HTTP link will use this information |
| Bonita Port | In the Email, the HTTP link will use this information |
| Email From | The 'mail from' attribute. Give a name, then the user can confirms this transmitter in its anti spam |
| Email Subject | Subject on the email |
| Email Text | Content of the email. Use the place holder {{assignedtasks}} and {{pendingtasks}} |
| Maximum tasks in the mail | Number of task to display in the email |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Time Execution | Time to execute the plug in (in ms) |
| Nb items processed | Number of items the job processed during the execution |

## Grumman Message correction

|  |  |
| --- | --- |
| Name | Grumman Message correction |
| Category | MONITOR |
| Job stopper | NO |

Execute the Grumman correction on message. See the Grumman page on Community for details.

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Only Detection | Only detection are executed. No correction. |
| Number of messages | Number of reconciliation messages treated at each execution |
| Reconciliation Incomplete | If true, then the Reconciliation Incomplete message are processed |
| Reconciliation Complete | If true, then the Reconciliation complete message are processed |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Time Execution | Time to execute the plug in (in ms) |
| Nb items processed | Number of items the job processed during the execution |

## Meteor

|  |  |
| --- | --- |
| Name | Meteor |
| Category | OTHER |
| Job stopper | MAXMINUTES |

Call a Meteor scenario

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Scenario name | Give the scenario name in Meteor |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Time Execution | Time to execute the plug in (in ms) |
| Nb items processed | Number of items the job processed during the execution |

## Monitor Directory

|  |  |
| --- | --- |
| Name | Monitor Directory |
| Category | MONITOR |
| Job stopper | BOTH |

Monitor a directory. When a file arrive in this directory, a new case is created

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Directory | This directory is monitor. Each file detected in this directory which match the filter trigger an action (create a case) |
| File Filter | File filter, to load only file who match the filter. \* and ? may be used.Visit https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html#sum |
| Execution | When a data is detected, which operation? Create a case: create a case, Execute a task: execute a task, Execute a message: execute a message. |
| Parameters to specify where the case will be created |  |
| Process Indentification | How to detect the process to create the case ? Static: process is specified. Dynamique: process is determined based on the file name, or a variable |
| Process name where case will be created | Give the process name to create case/execute tasks. Identify the process name  Give the process name. Expected result is [()]. Example, "ScanDirectory" or "ScanDirectory(1.4)" or "{{processNameInParameter}}" or "{{processNameInParameter}}({{processVersionInParameter}})" |
| Parameters to specify which tasks will be execution |  |
| Case ID | Give the way to find the caseId. Use {{}} to map to a data. Example: "{{caseId}}" |
| Task Name | Give the way to identify the taskname in the process. Use {{>}} to map a data. Example : "Review" or "{{task\_name}}" |
| Task ID | Instead of a caseId + TaskName, you can specify the way to retrieve the taskId. Use {{>}} to map a data. |
| Search Task by a Groovy code. | Give a Groovy code to search the Case Id. Use {{>}} to map a data, apiAccessor to access Bonita object. |
| Data Mapping | How to fullfill the contract for the execution |
| Bonita Contract value (JSON) | Give the contract, as JSON. {{}} may be used. Additionnal placeholder: {{FILESOURCE}}: content of the source, {{FILENAME}} FileName (not the directory), {{FILEDATEINMS}} date in millisecond. Example, "{ "invoiceInput": "{{FILESOURCE}}" } |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Time Execution | Time to execute the plug in (in ms) |
| Nb items processed | Number of items the job processed during the execution |

## Monitor Log

|  |  |
| --- | --- |
| Name | Monitor Log |
| Category | MONITOR |
| Job stopper | NO |

Analyze the log of the day before and checks errors

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Number of result in the TOP | Gives the number of line returned in the top x |
| Only errors in the TOP | If set, only error are returned in the TOP list |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Number of errors | Number of errors detected in the file |
| Number of Items detected | Number of errors items detected |
| Time Execution | Time to execute the plug in (in ms) |
| Total line in log | Number of lines in the logs |
| Number of different errors | One error can come multiple time. Analysis detect the duplicate to give a better situation |
| Nb items processed | Number of items the job processed during the execution |

## Move Case Archive

|  |  |
| --- | --- |
| Name | Move Case Archive |
| Category | CASES |
| Job stopper | BOTH |

Move cases to an archive database

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Datasource | Give the datasource name, where data will be moved |
| Filter on process | Job manage only process which mach the filter. If no filter is given, all processes are inspected |
| Delay to delete case, based on the Create date | All cases STARTED before this delay will be purged, even if they have a task execution after |
| Policy on Process Data | Process Data are copied to the Archive database, with the history or not |
| Policy on Local Data | Local Data are copied to the Archive database, with the history or not |
| History on process data | A process variable value may change in the process. Archive all value (or only the last value) |
| Policy on Process Data | Activities are copied to the Archive database. Light copied only one record per activity (else 3 to multiple records) |
| Policy on Document | Documents are copied to the Archive database |
| Policy on Comment | Comments are copied to the Archive database |
| External database | External database is a Archive data, or a different Bonita Instance. Attention, in the external Bonita Database, the process (name+version), username must exist. |
| Operation on case | Case are move (archived case will be detelete) or copy to the Archive database |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Time Execution | Time to execute the plug in (in ms) |
| Nb items processed | Number of items the job processed during the execution |

## Ping job

|  |  |
| --- | --- |
| Name | Ping job |
| Category | OTHER |
| Job stopper | MAXMINUTES |

Just do a ping

### Documentation

Ping Plugin is here for test usage, and explain to a developper how to creates its own plugin

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Add a date | If set, the date of execution is added in the status of execution |
| Time execution (in mn) | The job will run this time, and will update the % of execution each minutes |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Time Execution | Time to execute the plug in (in ms) |
| Random Value | Give a random value to demonstrate the function |
| Hour of day | Register the hour of day when the execution ran |
| Nb items processed | Number of items the job processed during the execution |

## Purge Archived Case

|  |  |
| --- | --- |
| Name | Purge Archived Case |
| Category | CASES |
| Job stopper | BOTH |

3 operations: PURGE/ GET LIST / PURGE FROM LIST. Purge (or get the list of) archived case according the filter. Filter based on different process, and purge cases older than the delai. At each round with Purge / Purge From list, a maximum case are deleted. If the maximum is over than 100000, it's reduce to this limit.

### Documentation

This Plugin has different usage.

Calculated a list of cases to purge

This option does not purge them, just calculate the list.

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 from List

> Then, you can delete case from a CSV list. This option works in partnership with the �Purge Archived Case/Get List� Plug-in.

It accepts the same list and check 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

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| operation: | Build a list of cases to operate, do directly the operation, or do the operation from a list  Result is a purge, or build a list, or used the uploaded list |
| Delay | The case must be older than this number, in days. 0 means all archived case is immediately in the perimeter |
| Process Filter | Give a list of process name. Name must be exact, no version is given (all versions will be purged) |
| Partial SubProcess purge | Root process information: Root Process are in the scope. In case of Purge, purge a root process purge all sub process information. Transient process information (sub process): Sub process information is in the scope, but not the root case. For example, you have a process FUNDTRANSFERT, call in a EXPENSE process; Reference the process FUNDTRANSFERT and select this option. All information about FUNDTRANSFERT called as a sub process is in the scope, but not but not EXPENSE or FUNDTRANSFERT called as a root process;Both: Both root and transient are in the scope. Then a FUNDTRANSFERT information call as root or as a sub process are in the scope |
| Separator CSV | CSV use a separator. May be ; or , |
| List of cases | List is calculated and saved in this parameter |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| cases purged | Number of case purged in this execution |
| Time Execution | Time to execute the plug in (in ms) |
| cases detected | Number of case detected in the scope |
| Nb items processed | Number of items the job processed during the execution |

## RGPD Policy

|  |  |
| --- | --- |
| Name | RGPD Policy |
| Category | OTHER |
| Job stopper | BOTH |

Purge data in the BDM, after a delay, to respect the RGPD

### Documentation

Missing documentation

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Bdm Object to clean | Give the BDM Object you want to purge. |
| BDM Date attribute | Attribute in the BDM with a Date. All object older than the delay are in the scope |
| Delay to operate the policy | All BDM Object older than this delay are in the scope |
| RGPD Policy | Purge some attribut, or delete the object |
| Attributes to clean | List all attributes to clean (separate by a comma, example FirstName,LastName,Age |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Time Execution | Time to execute the plug in (in ms) |
| Nb items processed | Number of items the job processed during the execution |

## Radar Bonita Engine

|  |  |
| --- | --- |
| Name | Radar Bonita Engine |
| Category | MONITOR |
| Job stopper | NO |

Monitor different indicators

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Radar Connector | Monitor the connector activity |
| Connector Too Long | Monitor connector execution. When an execution is too long, return it |
| Connector Limit Duration (seconds) | Report all connector execution longer than this value |
| Connector Frame (in minutes) | Search in all connector execution between now and now - frame |
| Radar Worker | Workers activity |
| Workers | Monitor workers. Return number of workers running, waiting |
| Statistics | Statistics activity |
| Process | Statistics on process (Number of process, number of active case) |
| Delay to collect processes deployed in this time frame | Collect processes deployed in this delay |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Time Execution | Time to execute the plug in (in ms) |
| Number of connectors with a long execution | Number of connectors with a long execution |
| Nb items processed | Number of items the job processed during the execution |
| Number of connectors called | Give the number of connectors called in the period |

## Replay Failed Tasks

|  |  |
| --- | --- |
| Name | Replay Failed Tasks |
| Category | TASKS |
| Job stopper | BOTH |

Monitor all failed tasks. Then after a delay, replay them, if the number of tentative is not reach

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Delay | Delay before asking to replay a failed task |
| Max tentatives | Maximum tentative to replay a failed task. After this number of tentative, job will not try to replay it. |
| Process Filter | List of processes in the perimeter. If no filter is given, all processes are in the perimeter |
| Number of tasks | Number of failed tasks detected, and replayed. |
| Only Detection | Only detection, do not replay tasks |
| Action on tasks : Replay or Skip | Task are replay, or skipped |
| Task and case in report | In the report, the list of Task/Case processed is saved |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Time Execution | Time to execute the plug in (in ms) |
| Nb items processed | Number of items the job processed during the execution |

## Replay (Stuck) Flow Nodes

|  |  |
| --- | --- |
| Name | Replay (Stuck) Flow Nodes |
| Category | TASKS |
| Job stopper | BOTH |

Check all flow nodes in the database, and if this number is over the number of Pending flownodes, restart them

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Delay consider task is stuck | In the report, the list of Task/Case processed is saved |
| Task and case in report | In the report, the list of Task/Case processed is saved |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Task execution error | Number of task in error when a re-execution was asked |
| Time Execution | Time to execute the plug in (in ms) |
| Tasks Executed | Number of task executed |
| Nb items processed | Number of items the job processed during the execution |

## SLA

|  |  |
| --- | --- |
| Name | SLA |
| Category | TASKS |
| Job stopper | NO |

Verify the SLA on human task, then notify if we reach some level like 75% of the task duration, of 100%. Can re-affect the task to a different user if needed.

ProcessName : if empty, then rule apply for all processes

Rule SLA

\* TASKNAME is the task where the rule apply (empty means all human task of the process)

\* PERCENT: 0 means the begining of the task. Else, calculation is perform between the start (0%) to the Due Date (100%). Over 100% is acceptable.

\* ACTION: (email use professionnal email)

EMAILUSER:

EMAILACTOR:

EMAILCANDIDATES

ASSIGNUSER: directly assign the task to a user.

ASSIGNSUPERVISOR:[1] assign to supervisor (1) or supervisor of supervisor(2) - stop if no supervisor.

STARTPROCESS:;;. JSONInput to match the input contract to start the process .

SENDMESSAGE:;;targetFlowNode;;

Place Holder Variable

When a JSON is required, theses place holder are available:

{{processName}} name of the process

{{processVersion}} version

{{processId}}: Process's ID

{{caseId}}: Case's ID

{{taskId}}: Task's ID

{{taskName}}: TaskName

{{percentThreashold}}: percent of SLA

{{dueDate}} : Due date

(example {'proc': '{{processname}}', 'taskId':{{taskId}}, 'SLA':'Level2' })Registration

To avoid to send twice a SLA email, you can setup a variable in the process. Job will register that it already sent the '60% milestone' email. Else, give a delay in minutes: if the milstone was between [CurrentTime - Delay, CurrentTime], then it is not resend

Documentation

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 these operations, based on �non interruptible boundary event�. When process have a lot of human tasks, and rules are important, the design may become very complex. Secondly, if a rule change (add a new trigger, change the trigger at 90 to 80%), the process must be redeployed.

This Plugin does these operations and can be update at any time.

The SLA operation is based:

A task�s name. If this parameter is empty, the rule is applied on all tasks

A Percent Threshold. This parameter 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 actions are possible. Click on detail Explanation to have a list of explanation. Main actions are EMAILUSER, EMAILACTOR, EMAILCANDIDATE,ASSIGNUSER, ASSIGNSUPERVISOR

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Process Name | Process name is mandatory. You can specify the process AND the version, or only the process name: all versions of this process is then checked |
| Rules SLA | Give a list of rules. Each rule describes the threshold in percent 0-task start, 100% due date, and action |
| Maximum tasks | In order to protect the server, this is the maximum task the rule will check in the process |
| Parameters mail | SLA Job may have a EMAIL action. This is then the email configuration. |
| Email from | The 'mail from' attribute |
| Email subject | Subject of the mail. See the content to the list of Place holder allowed |
| Email content | Content of the mail. Place holder are taskName, caseId, percentThreashold% threasold active, userFirstNameUser first name, receiver of the mail, userLastNameUser last name, receiver of the mail, userTitleUser title, receiver of the mail |
| Bonita server | HTTP address to join the caseId, if you place the taskURL in the content on the mail, to access the Bonita server |
| Rebound mechanism | To avoid to execute twice an action on a task, different mechanism are available. |
| Process Variable registration | To avoid to execute the same rule, give access to a Process Variable HASHMAP, then all executions are registered on this data |
| Delay registration in minutes | To avoid to execute the same rule, and you can't give a Process variable, allow a registration in minute. Action is executed if the time slot is 'Fire < Current time < File + Duration' |
| SMTP Parameters | To send an email, SMTP parameters is required to connect a SMTP server |
| SMTP Parameters | Smtp parameters: host:port:username:password |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Time Execution | Time to execute the plug in (in ms) |
| Nb items processed | Number of items the job processed during the execution |

## Sonar

|  |  |
| --- | --- |
| Name | Sonar |
| Category | PROCESSES |
| Job stopper | BOTH |

Call a Sonar verification on processes

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Process Filter | Give a list of process name. Name must be exact, no version is given (all versions will be purged) |
| Process Version Policy | Manage all processes, or only the last version (last deployed) |
| Process Status Policy | Manage all processes, or only ENABLE, DISABLE process |
| Delay | All processes deployed inside this delay are checked |
| When a process has errors | When a process has error, it can be disabled |
| Parameters | Use the default parameter, or the one saved in the Sonar Page |
| Qualification | Use the default qualification, or the one saved in the Sonar Page |
| Report Level | Select the level reported. If INFO, only INFO, TIP, WARNING, SEVERE are reported for example |
| Report Sonar | Result of Sonar Analysis |
| CsvSeparator | The CSV Separator is defined in the Sonar Parameters. Access this page to change it |

### Measure

|  |  |
| --- | --- |
| Name | Explanation |
| Number of errors | Give the number of errors detected |
| Number of processes in errors | Give the number of process in error detected (ERROR) |
| Time Execution | Time to execute the plug in (in ms) |
| Number of processes correct | Give the number of process correct detected (WARNING,INFO,TIPS) |
| Ratio process correct | Give the ratio (in %) of correct process |
| Nb items processed | Number of items the job processed during the execution |

## Unassign Tasks

|  |  |
| --- | --- |
| Name | Unassign Tasks |
| Category | TASKS |
| Job stopper | BOTH |

Unassign tasks if not resolved after a specified time.

### Parameters

|  |  |
| --- | --- |
| Name | Explanation |
| Process Name | Process name is mandatory. You can specify the process AND the version, or only the process name: all versions of this process is then checked |
| Tasks Name | The task name to search for. If empty, all tasks in process are concerned. A list of tasks, separate by , can be provided too. |
| Checkout time execution (in mn) | How many minutes until the item should be unassigned |

### Measure

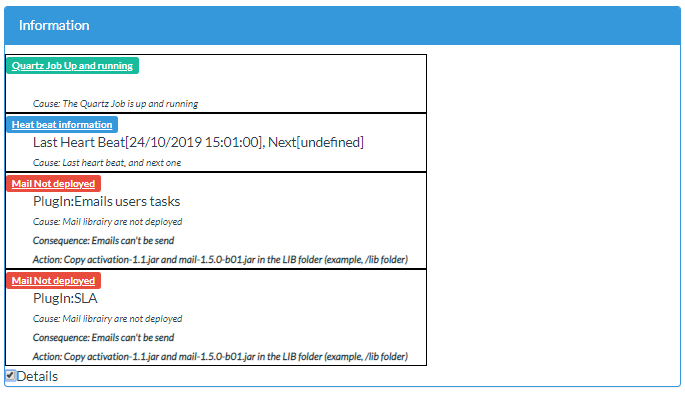
|  |  |
| --- | --- |
| Name | Explanation |
| Time Execution | Time to execute the plug in (in ms) |
| Nb items processed | Number of items the job processed during the execution |

# Maintenance

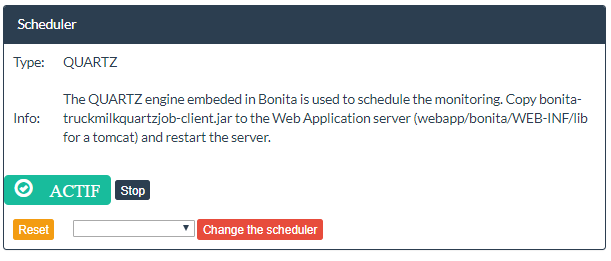
The Maintenance tab group some operations:

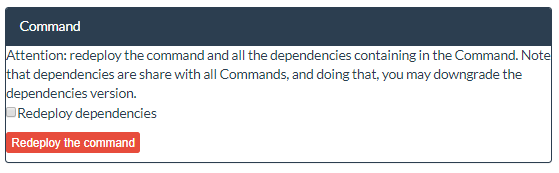
## Information

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. Then user can create a new job from your Plug-in.

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

The convention is to have a sub package per CATEGORY. The category explains on which component the job works: it may be tasks, cases, monitor...

You can set the type to TYPE\_PLUGIN.EMBEDED : that's mean this Plug-in is delivered with the Truckmilk page.

1. Register your plug in in MilkPlugInFactory.collectListPlugIn()

Then, the factory knows your Plug-in and can propose it to the users

1. Define the getDefinitionDescription()

This method returns the definition of your Plug-in parameters, category, measures.

1. Define the execute() method

Now you are! this is the main part of your Plug-in, the execution.

At input, Truckmilk gives you the value of all parameters the user gives. This is the value for parameters defined in the getDefinitionDescription().

At the output, you must produce a status (SUCCESS, ERROR or else), reports and measures.

1. Additional methods: checkPluginEnvironment() and checkJobEnvironment()

If you want, you can implement this method. The checkPluginEnvironment() is called at the beginning, first time your Plug-in is registered.. You may want to test if all additional information is set. For example, you may need an external JAR file: it is present?

checkJobEnvironment() is different. It depends of the job definition, and it is executed before each execution. For example, you ask as a parameter a

DataSource to connect to an external database. Is this database can be accessed before the execution

## Use Bonita Events

TruckMilk use the BonitaEvent library to display event (which may be information or error)

It's very important that the user describe the error, explain it, and give an action plan. Instead to give a SQL ERROR to the user, developers has to explain what's happen, what is the consequence, and what is the action to do.

For each event, you must specify:

* a package and a number. Then, the event is unique and can be identified immediately by the administrator and the developer to figure out what's going on
* a level (INFO, ERROR, WARNING, CRITICAL)
* a title to display to the user. Example "Bad SQL Request"
* a cause: it's mandatory for an error. The SQL Request failed: what is the cause? Can't connect to the database? Bad syntax? Missing parameters?
* a consequence: it's mandatory for an error: explain to the user the consequence of this error. Ok, the SQL can’t be executed, so we have no history
* for example, but the treatment can continue.
* an action: what the user has to do? Database is done, then action is to contact the administrator of the database behind the source.

Then, you can create a new event, referencing an existing event. Doing that, you can add parameters.

**private** **static** BEvent *eventDeletionFailed* = **new** BEvent(MilkMoveArchive.**class**.getName(),

2,

Level.***ERROR***,

"Error during deletion",

"An error arrived during the deletion of archived cases",

"Cases are not deleted",

"Check the exception");

milkJobOutput.addEvent(**new** BEvent(*eventDeletionFailed*, e, "ListArchiveId: " + listArchivedProcessInstances.toString() + " " + e.getMessage()));

## Check Plug-in environment

Plug-in can check its environment, to detect if you missed something.

**public** List<BEvent> checkPluginEnvironment(MilkJobExecution milkJobExecution)

An external component may

* be required and are not installed.
* This call is performed when users click on "getStatus" in the maintenance panel, and before each execution.

**@return** a list of Events.

## Check Job environment

Check the Job's environment.

**public** List<BEvent> checkJobEnvironment(MilkJobExecution milkJobExecution)

This verification is executed before each execution, and all execution parameters is given at input. So, the verification is done with the real input.

For example, the Plug-in asks a Data source as parameters. So, it can check in this method that it can connect to the database.

## Definition description

The method returns the description the Plug-in.

**public** MilkPlugInDescription getDefinitionDescription(MilkJobContext milkJobContext)

The description contains:

* a name (which must be unique), explanation and label

milkPlugInDescription.setName("Ping");

milkPlugInDescription.setExplanation("Just do a ping");

milkPlugInDescription.setLabel("Ping job");

* a category

Choose between the existing category

milkPlugInDescription.setCategory(CATEGORY.***OTHER***);

* the way to ask to stop the job. Users can decide to limit the job execution in time, or in number of items processed. Of course, if you describe you can stop in time, you have to implement it in your code.

milkPlugInDescription.setStopJob(JOBSTOPPER.***MAXMINUTES***);

* the list of parameters

milkPlugInDescription.addParameter(***cstParamAddDate***);

milkPlugInDescription.addParameter(***cstParamTimeExecution***);

* the list of measures

milkPlugInDescription.addMesure(***cstMesureMS***);

milkPlugInDescription.addMesure(***cstMesureHourOfDay***);

## Using parameters

A Plug-in needs parameters in general. To delete cases, the list of processes, or a delay to collect only cases older than a delay for example.

You register parameters in the getDefinitionDescription(), and you use it in the checkJobEnvironment() or in executeJob().

You have different kind of parameters. Then the parameters are visible in the Parameters page, when users access this section.

You define a parameters with the createInstance() method.

**private** **final** **static** PlugInParameter ***cstParamAddDate*** = PlugInParameter.*createInstance*("addDate", "Add a date", TypeParameter.***BOOLEAN***, **true**, "If set, the date of execution is added in the status of execution");

Different parameters are:

* Th name of the parameter (should be unique in all Plug-in).
* The label user will see.
* The type. See below to have an information on each type
* The default value. The default value depends of the type (must be a Boolean is you choose a type parameter BOOLEAN)
* Then, a complete information. When user clicks on “show information”, he will see that.

You have some shortcut, like

* createInstanceDelay(),
* createInstanceArrayMap(),
* createInstanceButton,
* createInstanceFile,
* createInstanceListValues,
* createInstanceInformation

Then, you access the value of parameters via the getInput….Parameters method

Boolean addDate = milkJobExecution.getInputBooleanParameter( ***cstParamAddDate*** );

Here the different type parameters

### Basic type

In this category, A STRING is a simple input, and a TEXT a Text area, a LONG a number, a BOOLEAN a check box

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | |
| STRING | | A simple input |
| TEXT | | A Test Area input |
| LONG | | An input, type number (only number can be gives) |
| BOOLEAN | | A Checkbox |

### Complex type

These parameters handle complex display

|  |  |
| --- | --- |
| **Type** | **Description** |
| DELAY | User give a delay. A selection allows the user to give a type (Minutes, Hours, Days, Month) and a second input the value. Then user can give “4 days” or “10 hours”.  As a developer, you give a date, and a direction, and you get back the calculated day. Example, you give “February 1, 10:00” + “advance”, for a 4 days delay, you’ll receive “February 5 10:00”. |
| LISTVALUES | A selection box |
| ARRAY | A List of String. User can add / remove items |
| ARRAYMAP | A list of Records. Record is defined as a list of “ColDefinition”.  Plug-in get a List of Map: List<Map<String,Object>> |
| JSON | An input text, where the user is supposed to give a JSON string. The JSON is verified. |

Example of ARRAYMAP definition

PlugInParameter *cstParamRuleSLA* = PlugInParameter.*createInstanceArrayMap*("RuleSLA", "Rules SLA",

Arrays.*asList*(

ColDefinition.*getInstance*("TASKNAME", "TaskName", "Task name in a process. Multiple tasks may be reference one time using # like 'Validate#Review#Control", TypeParameter.***STRING***, 50),

ColDefinition.*getInstance*("PERCENT", "Percent Threashold", "0%: task creation, 100%=due date", TypeParameter.***LONG***, 20),

ColDefinition.*getInstance*("ACTION", "Action", ACTION.***EMAILUSER***.toString() + ":<userName>, "

+ ACTION.***EMAILCANDIDATES***.toString() + ", "

+ ACTION.***EMAILACTOR***.toString() + ":<actor>,"

+ ACTION.***ASSIGNUSER***.toString() + ":<userName>, "

+ ACTION.***ASSIGNSUPERVISOR***.toString() + ":[1], "

+ ACTION.***STARTPROCESS***.toString() + ":<processName (<processVersion)>;<JSONinput>,"

+ ACTION.***SENDMESSAGE***.toString() + ":<messageName>;<targetProcess (<processversion>);<targetFlowNode>;<JSONMessageContent>;<JSONMessageCorrelation>",

TypeParameter.***TEXT***, 50)),

**null**, "Give a list of rules. Each rule describes the threshold in percent 0-task start, 100% due date, and action");

### Documents type

Manipulate document, as input or at output.

|  |  |
| --- | --- |
| **Type** | **Description** |
| FILEREAD | User upload a document, and Plug-in can read it. |
| FILEWRITE | Plug-in write this document, and it will be available for the user as download |
| FILEREADWRITE | User can upload a document, Plug-in can read, and write a new version |

### Bonita type

These types display some Bonita Artifact

|  |  |
| --- | --- |
| **Type** | **Description** |
| USERNAME | Autocomplete to select a user between all users in the server |
| PROCESSNAME | Autocomplete to select a process in all processes (ENABLE or DISABLED) in the server |
| ARRAYPROCESSNAME | Autocomplete to select one or multiple processes |

### Special usages

|  |  |
| --- | --- |
| **Type** | **Description** |
| BUTTONARGS | Use to execute a local simulation |
| SEPARATOR | Separators, when you have a lot of parameters |
| INFORMATION | Display information |

## Conditions with parameters

Some conditions can be added on parameters

For example, you can use

**private** **static** PlugInParameter *cstParamProcessFilter* = PlugInParameter.*createInstance*("processfilter", "Filter on process", TypeParameter.***ARRAYPROCESSNAME***, **null**, "Job manage only process which mach the filter. If no filter is given, all processes are inspected")

.withMandatory(false)

.withFilterProcess(FilterProcess.***ALL***);

withMandatory( boolean isMandatory) :

|  |  |
| --- | --- |
| **Type** | **Description** |
| withMandatory( boolean isMandatory) | Field is mandatory. |
| withVisibleCondition( String visibleCondition ) | Give a condition. Attention, condition is in JavaScript |
| withVisibleConditionParameterValueDiff( PlugInParameter parameter, Object value ) | Give a condition based on a value of a different attribute. |
| withVisibleConditionParameterValueEqual( PlugInParameter parameter, Object value ) | Give a condition based on a value |
| withFilterProcess(FilterProcess filterOnProcess)  enum FilterProcess { ALL, ONLYENABLED, ONLYDISABLED } | The process displayed in the parameters are only ENABLE, or DISABLE according the filter |

Example on Visible Condition

**private** **static** PlugInParameter *cstParamOperation* = PlugInParameter.*createInstanceListValues*("operation", "operation: Build a list of cases to operate, do directly the operation, or do the operation from a list",

**new** String[] { ***CSTOPERATION\_GETLIST***, ***CSTOPERATION\_DIRECT***, ***CSTOPERATION\_FROMLIST*** }, ***CSTOPERATION\_DIRECT***, "Result is a purge, or build a list, or used the uploaded list");

**private** **static** PlugInParameter *cstParamDelay* = PlugInParameter.*createInstanceDelay*("delayinday", "Delay", DELAYSCOPE.***MONTH***, 3, "The case must be older than this number, in days. 0 means all archived case is immediately in the perimeter")

.withMandatory(**true**)

.withVisibleConditionParameterValueDiff(cstParamOperation, CSTOPERATION\_FROMLIST);

This parameter “delayinday” is visible if the choice in operation is different than “FROMLIST”

Example on Filter process:

**private** **static** PlugInParameter *cstParamProcessFilter* = PlugInParameter.*createInstance*("processfilter", "Process Filter", TypeParameter.***ARRAYPROCESSNAME***, **null**, "Give a list of process name. Name must be exact, no version is given (all versions will be purged)")

.withVisibleCondition("milkJob.parametersvalue[ 'operation' ] != '" + ***CSTOPERATION\_FROMLIST*** + "'")

.withFilterProcess(FilterProcess.***ALL***);

## Execution

Execution is the main part of the Plug-in.

**public** MilkJobOutput executeJob(MilkJobExecution jobExecution) {

The Plug-in gets the value on each parameter and produce a result.

**Input**

Different objects are accessible via the milkJobExecution.

* all parameters are accessible
* an APIAccessor are accessible too.

Example:

String operation = milkJobExecution.getInputStringParameter(*cstParamOperation*);

DelayResult delayResult = milkJobExecution.getInputDelayParameter( *cstParamDelay*, **new** Date(), **false**);

ProcessAPI processAPI = milkJobExecution.getApiAccessor().getProcessAPI()

**Output**

You have multiple mechanism.

* executionStatus: the status must be given to milkJobOutput.executionStatus.
* number of item processed: use setNbItemsProcessed() to give the number of item processed.
* listEvents: the detail of execution is provided by a list of Events. Use milkJobOutput.addEvent()
* Measure: you can add all measure in the report too, else they are in the measure table (user need to access it separately)
* Chronometers: add the chronometers information’s.
* addReportInHtml() to add a HTML sentence
* addReportTableBegin() / addReportTableLine() / addReportTableEnd() : give data to safe it in HTML. Table has a protection to not keep a big amount of
* Document output

## Measures

A measure is a number that you want to calculate and return.

Measure is saved in a different table, and you can save only the two last report execution, but keep 1000 measures, to display a graph.

Two measures are automatically saved: the number of items processed and the time of execution.

First, you have to declare the measure in the Description:

PlugInMeasurement cstMesureMS = PlugInMeasurement.createInstance( <Code>, <Label>, <Explanation>);

milkPlugInDescription.addMesure( cstMesureMS );

Then in the execution, you can set a value to the measure

milkJobOutput.setMeasure( cstMesureMS, <value>);

If you want to add all measure in the report, use

milkJobOutput.addMeasuresInReport(boolean keepMesureValueNotDefine, boolean withEmbededMeasure);

keepMesureValueNotDefine if true, a measure not defined in the execution is added in the report, with the value 0

withEmbededMeasure Some measure are embedded: number of items processed and time to execute. They can be added in the report.

## Chronometers

For a long execution time, you want to register the time to execute a piece of code. Then, theses value can be added in the final report

To Start a chronometer, use

Chronometer sleepTimeMarker = milkJobOutput.beginChronometer( <ChronometerName> );

To stop it, use

milkJobOutput.endChronometer( sleepTimeMarker);

A Chronometer save the time from Begin to End, and the number of occurrences. So, you get the final information on the total number of executions, and the number of occurrences.

Add all chronometers in the final report by

milkJobOutput.addChronometersInReport(addNumberOfOccurrence, addAverage);

addNumberOfOccurrence if true, the number of occurrences is added in the report

addAverage if true, the average is added in the report (total time / numberOfOccurence)

## Advancement

To give a feedback to the user, an advancement can be calculated and send back to the user.

Note: the stop mechanism can stop immediately the job, and then the advancement may be stuck at final at 45% for example

You have two ways to give back a status to user.

milkJobOutput.setAvancementTotalStep( <totalStep>)

milkJobExecution.setAvancementStep(<StepValue>)

You setup the total number of steps you detect. Imagine that you want to delete cases. You detect that you have to delete 454 cases. Then, set the totalNumber to 454, and after each deletion, set the avancementStep to the number of case deleted.

Or use

milkJobOutput.setAvancement( advancementInPercent )

It's up to you to calculate the % of advancement. To give more feedback on what's is going on, use the milkJobOutput.setAvancementInformation(String information) to set some information.

Please note:

Truckmilk does not update in the database the advancement at each time. it does that only every X second (30 seconds). This is to avoid slowing down the performance. Imagine that you set the TotalStep to 4 412 044, and you update the advancement for each step. You don't want that Truckmilk do 4 412 044 updates in the database. So, if the last setAvancement() was done in the last 30 seconds, it register the value in memory only.Same for the percentage: if the percentage does not change, there is no update in the database

## Stop mechanism

If your treatment take time, you have to implement a stop mechanism.

Truckmilk will not kill your thread, in order to not abort brutally a treatment. So, you can choose when you can stop. Just check the method:

milkJobExecution.isStopRequired()

If it is true, then the stop is required.

For your information, there are three way to stop:

* user explicitly require it by clicking on the stop button
* user specify a "maximum execution time" and your execution reach this limit
* user specify a "maximum item to process", and, via the setNbItemsProcessed(), you reach this number

## Document management

Your execution may need to access a document or will create a document. The report is limited in size, so if you need to produce a list of information, you should not use the Report, but produce a document (a CSV file for example).

A document can be accessed in READ, in WRITE (you produce it) or in READ/WRITE (you update a list of information).

First, you have to declare the parameter. You must give a file name and a content type (used by the browser when you upload the document)

PlugInParameter cstParamMyReadDocument = PlugInParameter.createInstanceFile("readIt", "Read the document", TypeParameter.FILEREAD, null, "Read", "Read.csv", "application/CSV")

PlugInParameter cstParamMyWriteDocument = PlugInParameter.createInstanceFile("writeIt", "Write the document", TypeParameter.FILEWRITE, null, "Write", "Write.csv", "application/CSV")

PlugInParameter cstParamMyReadWriteDocument = PlugInParameter.createInstanceFile("readWriteIt", "Read-Write the document", TypeParameter.FILEREADWRITE,

\* null, "List is calculated and saved in this parameter", "ReadWrite.csv", "application/CSV")

Plug-in reads the document by:

List<BEvent> milkJobExecution.getParameterStream(<PlugInParameter>, <OutputStream>)

All the content is sent to out OutputStream.

Plug-in writes the document by:

milkJobOutput.setParameterStream( <PlugInParameter>, <InputStream>)

Note: documents are store in the database as a BLOB.

## State of the art

* Don’t hard code any value in the Plug-in. Use parameters. For example, if your Plug-in has 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 parameter.
* 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, Plug-in 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 Plug-in has to be robust. 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