Meteor Manual

Tests

June 2018

Sommaire

[1. Introduction 3](#_Toc489972090)

[2. Usage 3](#_Toc489972091)

[3. Installation 3](#_Toc489972092)

[4. Build a scenario 3](#_Toc489972093)

[4.1. Process 3](#_Toc489972094)

[4.2. Command Scenario 6](#_Toc489972095)

[4.3. Groovy Scenario 7](#_Toc489972096)

[4.4. Use an external server 7](#_Toc489972097)

[5. Export / Import the scenario 7](#_Toc489972098)

[6. Run a scenario 7](#_Toc489972099)

[6.5. Manually 7](#_Toc489972100)

[6.6. By a REST CALL 7](#_Toc489972101)

[6.6.1. Start 7](#_Toc489972102)

[6.6.2. Status 8](#_Toc489972103)

[6.7. Inside a CI environment 8](#_Toc489972104)

Content

# Introduction

This document explains how to use the Meteor Page.

# Usage

The page may be use in different context:

* To advance a case at the task
* To realize a load test
* To run a Regression and performance test, in a CI environment

# Installation

The Meteor page is a Bonita custom page. To install it:

* From the Foodtruck, click on the icone and download it
* Directly form the Bonita Community or the Github project. Then as an administrator, go to the Bonita Portal / resource and import the ZIP file

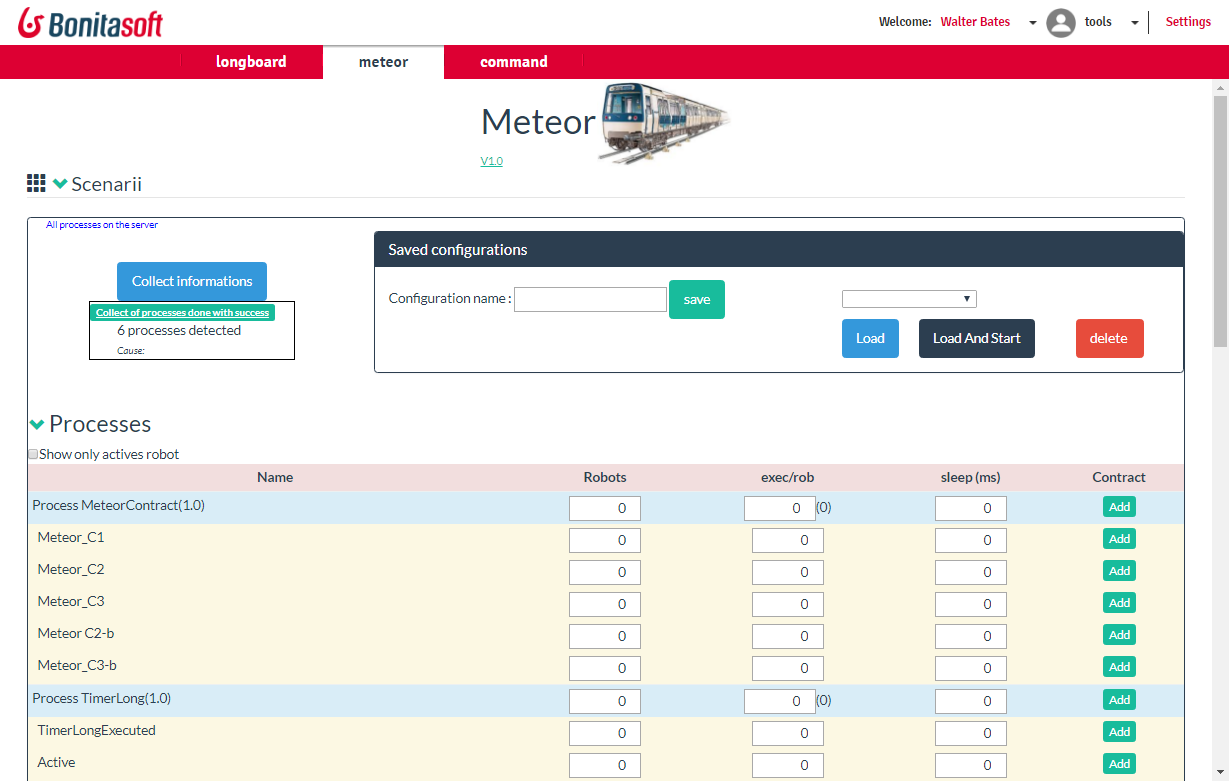
Then, register the page in a Profile or in an Application (Community version).

# Build a scenario

## Process

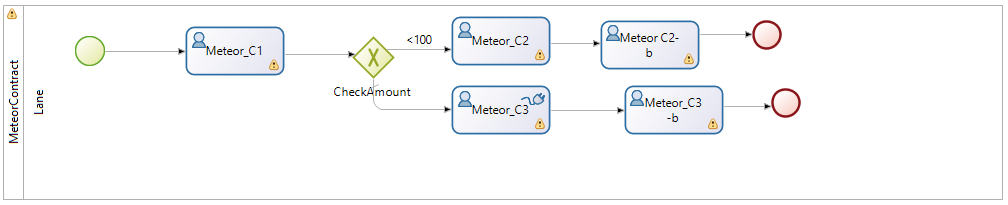
At startup, or when you click on the “Collect information” button, the page collect all information on the server.

All existing processes are displayed.



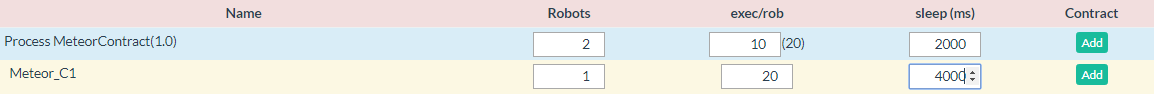
For each process, each human task, you can setup a “robot”. A robot simulates an external operation to create a case (on the process level) or to execute a task.

This is the process to test:

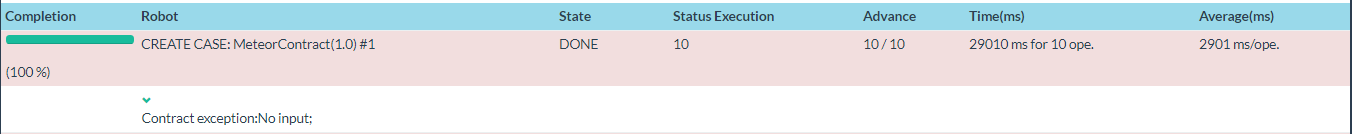


To create some case, we want to start 2 robots, to load the server. Each robot will create 10 cases (so 2\*10=20 cases created at end) and between each creation, robot will sleep 2000 s to simulate the existing system.

When cases are created, they will move to the task “Meteor\_C1”, a human task. The meteor page will execute the task, simulating a user. We want to simulate only one user, to 1 robots. There are 20 cases, so let’s execute the task 20 times.

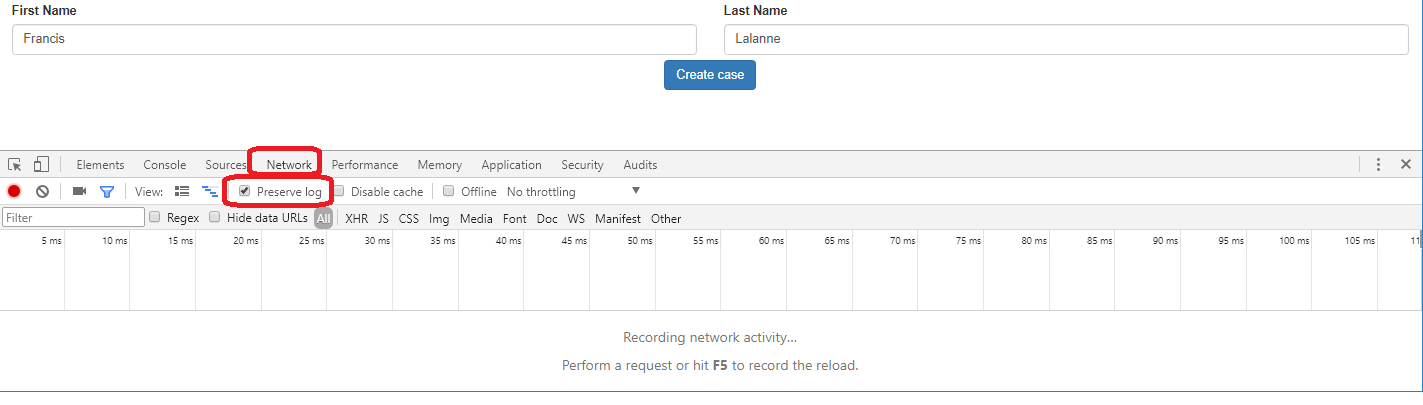


If you start the process as it, all creation will have failed because the process creation expect an Input contract



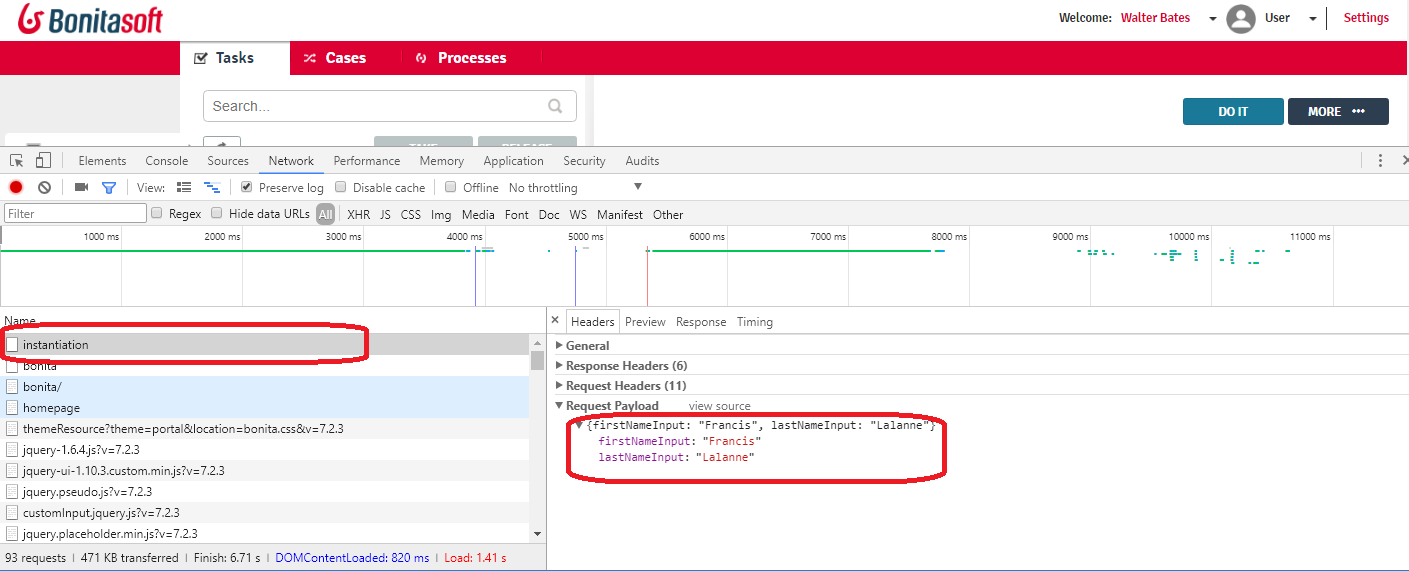
To figure out the input to give, you can get it from the browser.

In Chrome, access the form, and then click on F12. Then, select the Network. Click the check box “preserve log”.



Click on the button to create a case.

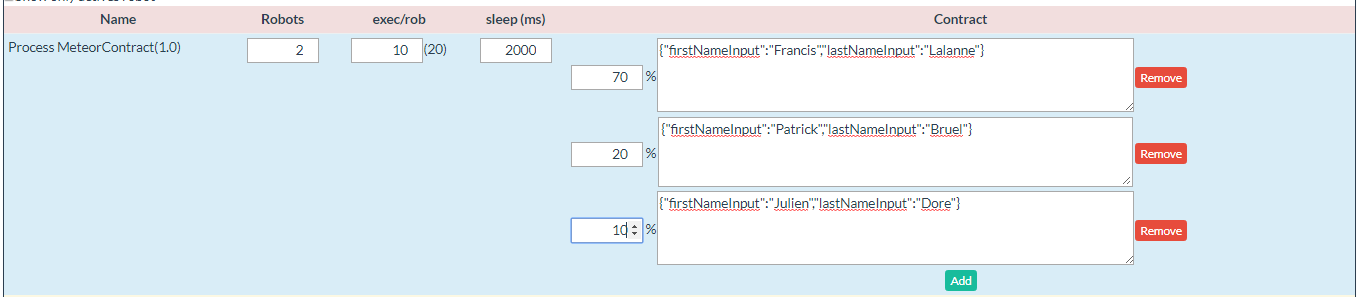
Then, search the Instantiation URL (should be the first)



The RequestPayload is the information you look for (click on “view source” to get the correct one): here it’s

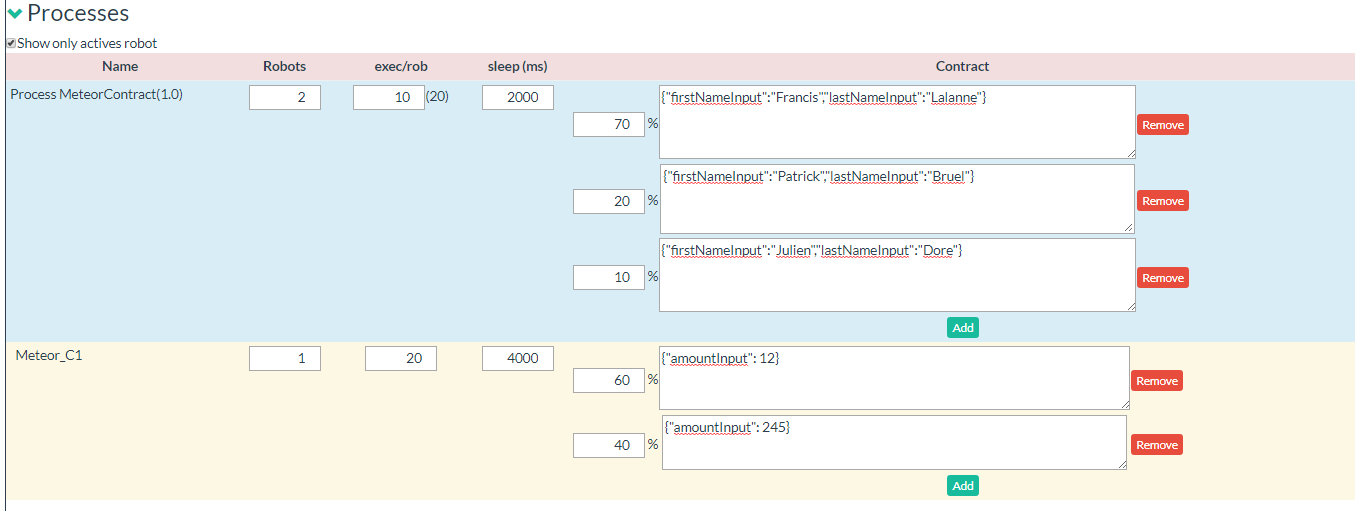
{"firstNameInput":"Francis","lastNameInput":"Lalanne"}

So, we can now create the test game



We set three different value, with a percent of usage. In 70% of the creation, the first test game is used; in 20% the second one. In fact, the total could not do a 100%, the page will use the different value to calculate the number of case. for example, give “7”, “2”, “2” is correct too, ratio will be calculated on a base 11 (7+2.2).

We do that on the first human task too



Your scenario is ready

## Command Scenario

The command scenario is a list of sentence. One execution run all the different sentence.

The sentence can be give in two different ways:

* using a language with a list of verb
* by a complete JSON format

The JSON format is useful to reduce the syntaxe error (then, use an external tool like <http://www.jsoneditoronline.org/> to edit the JSON).

The different verb are:

|  |  |  |
| --- | --- | --- |
| **Sentence** | **Json** | **Explanation** |
| createCase( "processName": "myProcess",  "processVersion": "1.0",  "nbExecution": 10,  "input": {"firstnameInput":"walter" }); | {  "verb": "createCase",  "processName": "MeteorContract",  "processVersion": "",  "nbExecution": 10,  "input": {  "firstNameInput": "Walter",  }  } | Create a case in a process |
| executeTask("processName": "MeteorContract",  "taskName": "Meteor\_C1",  "nbExecution": 8,  "input": {"amountInput": 12 } ); | {  "verb": "executeTask",  "processName": "MeteorContract",  "taskName": "Meteor\_C1",  "nbExecution": 8,  "input": {  "amountInput": 12  }  } | Execute a task |
| Sleep( 3000 ); | {  "verb": "sleep",  "timeSleep": "30000"  }, | Sleep time |

## Groovy Scenario

## Use an external server

To be implemented

# Export / Import the scenario

To be implemented

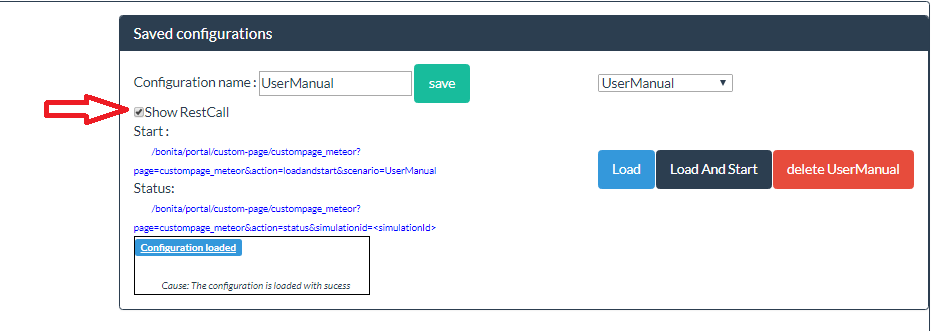
# Run a scenario

## Manually

## By a REST CALL

Two REST API is available to run a RestCall. Select a configuration, and then click on “load”.

Click on the check Box “ShowRestCall” to access them



### Start

The RestAPI start is used to start a new test. The form is

<ServerBonita>/bonita/portal/custom-page/custompage\_meteor?page=custompage\_meteor&action=loadandstart&scenario=<Config name>

Nota: if the name is complexe and contains special character, you can use the JSON form (which is the URL use by the page) :

[http://localhost:8080/bonita/portal/custom-page/custompage\_meteor/?page=custompage\_meteor&action=loadandstart&paramjson={%22confname%22:%22<ConfigName>%22}](http://localhost:8080/bonita/portal/custom-page/custompage_meteor/?page=custompage_meteor&action=loadandstart&paramjson=%7b%22confname%22:%22%3cConfigName%3e%22%7d)

To create this URL, use this JS script

**var** param = { "confname": **this**.config.currentname};

**var** json = encodeURI( angular.toJson( param, **false**));

$http.get( '?page=custompage\_meteor&action=loadandstart&paramjson='+json )

The resultis a JSON information, which contains two main information:

* The status
* The simulationID

{

"timeStarted": "08/08/2017 11:08:33",

"robots": [

...

],

"config": {

....

},

"simulationid": "1502217803974",

"status": "STARTED"

}

You use the simulationId to get the status

### Status

To get the status of a simulation, use

<Server>/bonita/portal/custom-page/custompage\_meteor?page=custompage\_meteor&action=status&simulationid=<simulationid>

Result is a JSON loke

{

"timeEnded": "08/08/2017 12:08:54",

"timeStarted": "08/08/2017 12:08:33",

"robots": [

...

],

"listSimulations": [

...

],

"nbErrors": 0,

"status": "DONE"

}

The different status are :

* DONE : the test is finish
* NOSIMULATION : the simulation id given not exist
* STARTED: the simulation is in progress

When the simulation is running (state STARTED), you have some additional attributes

percentAdvance is the percentage of advancement( 0 to 100)

## Inside a CI environment

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