



# DIATEAM

MILITARY & CIVIL CYBERSECURITY  
SOLUTIONS

## Action Manager Front Documentation 2.6.0 version

Référence: AM-DOC-FRONT-2.6.0  
Référence(s) auteur(s): DIATEAM  
Version: 1.0

# TABLE OF CONTENTS

<b>1</b>	<b>System presentation</b>	<b>3</b>
1.1	Context . . . . .	3
1.2	Software design . . . . .	3
1.3	Integration into hynesim . . . . .	3
<b>2</b>	<b>Virtual machine setup</b>	<b>4</b>
2.1	Hynesim setup . . . . .	4
2.2	Guest setup . . . . .	4
2.2.1	Files location . . . . .	4
2.2.2	Linux stub installation . . . . .	5
2.2.3	Windows stub installation . . . . .	6
<b>3</b>	<b>Usage</b>	<b>8</b>
3.1	Concepts . . . . .	8
3.2	Basic features . . . . .	8
3.2.1	List topologies . . . . .	8
3.2.2	List agents . . . . .	8
3.2.3	List runs . . . . .	9
3.2.4	List actions . . . . .	10
3.2.5	Actions versioning . . . . .	11
3.2.6	Get action details . . . . .	11
3.2.7	Create actions . . . . .	11
3.2.8	Import actions . . . . .	13
3.2.9	Download actions . . . . .	13
3.2.10	Run actions . . . . .	15
3.2.11	Purge runs . . . . .	16
3.2.12	Remove entities . . . . .	17
3.3	Advanced features . . . . .	18
3.3.1	Scheduler . . . . .	18
3.3.2	Action inputs . . . . .	19
3.3.3	Action resources . . . . .	22
3.3.4	Action outputs . . . . .	23
3.3.5	User actions . . . . .	25
3.3.6	Executable name . . . . .	26
3.3.7	Executable path . . . . .	26
3.4	Error management . . . . .	27
3.4.1	Windows Powershell . . . . .	28
3.4.2	Windows batch . . . . .	29
3.4.3	Linux . . . . .	29

# 1. System presentation

## 1.1 Context

The goal of the Action Manager is to be able to remotely run actions inside hynesim entities. This framework enables users to develop their own actions to instrument virtual machines. It can be used to perform automatic actions, such as:

- Loading a webpage
- Running a program inside of virtual machines
- Reading or writing files

With a set of actions, one can launch multiple actions, fetch their results and react to their outputs to automatically play scenarios inside topologies, which could be:

- Malicious file or attachment opening by a user
- Attack scenario (vulnerability scanning, exploitation...)
- Generating background traffic

## 1.2 Software design

The agent is a software client running on the instrumented virtual machine. Its role is to contact the host upon startup, to fetch available actions and run them. The host acts as a server and runs on the hynesim master. It is the endpoint for client interaction. Its goal is to manage agents, handle client requests and transmit orders to agents. It also stores input and output files, actions and agents history. The clients are users or programs that communicate with the host using its REST interface. Communicating with the REST interface can be done through classic URL requests or using the front web application.

## 1.3 Integration into hynesim

The Action Manager agent communicates over a serial port which has to be activated on the virtual machine.

## 2. Virtual machine setup

### 2.1 Hynesim setup

Open the virtual machine properties on which you wish to use the Action Manager. Go to the **Hynesim** tab and check **Serial port for action manager**.

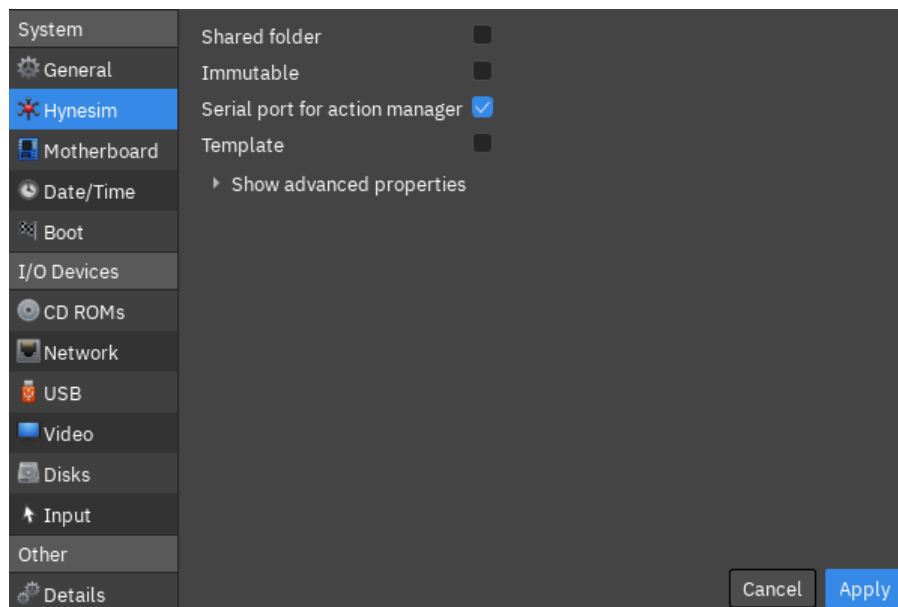


Fig. 1: Serial port activation in hynesim

### 2.2 Guest setup

#### 2.2.1 Files location

Get the stub binary through the Action Manager API using this URL:

`{APIURL}/v2/stub`

- {APIURL} is the Action Manager API URL (usually <http://am.hns-platform.com>)

There is an easy access from the web application with the **tools** button located at the top right corner of any page, near the API documentation.

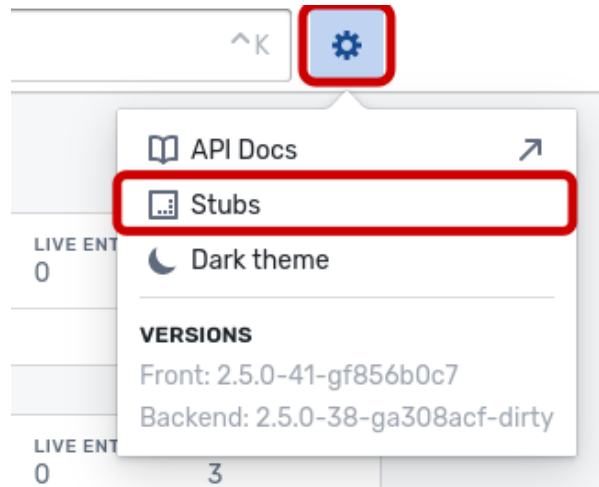


Fig. 2: **Stub** button for resources

Then you can choose on the left if you want a binary for Windows or Linux. On each pages you have the instructions to install the stub on the target VM.

## 2.2.2 Linux stub installation

### Binary and config

In the virtual machine, create a *hns-actionmanager* folder as root at the location */opt*. Put the stub file in this folder, resulting in the path */opt/hns-actionmanager/stub\_linux-amd64*.

Finally make it executable by running:

```
chmod +x stub_linux-amd64
```

### Installation

To install the stub, run:

```
./stub_linux-amd64 --install
```

This will create a new *am-stub* service that will be started and enabled (meaning it will start at every machine startup).

### Troubleshooting

If the service does not appear to be running, check:

- If the stub is executable
- If the serial port box is checked in hyview

## 2.2.3 Windows stub installation

### VirtIO setup

First, you need to install VirtIO drivers to allow for a serial connection in the machine. Download the latest stable version [here](#) and put it in your guest machine.

Double-click on the ISO and run the installer fitting your architecture.

Name	Date modified	Type	Size
amd64	7/20/2020 7:59 PM	File folder	
Balloon	7/20/2020 7:57 PM	File folder	
data	7/20/2020 7:59 PM	File folder	
guest-agent	7/20/2020 8:04 PM	File folder	
i386	7/20/2020 7:59 PM	File folder	
NetKVM	7/20/2020 7:57 PM	File folder	
pvpanic	7/20/2020 7:57 PM	File folder	
qemu_fwcfg	7/20/2020 7:57 PM	File folder	
qemu_pcserial	7/20/2020 7:57 PM	File folder	
qxl	7/20/2020 7:57 PM	File folder	
qxldod	7/20/2020 7:57 PM	File folder	
smbus	7/20/2020 7:57 PM	File folder	
vioinput	7/20/2020 7:57 PM	File folder	
viornig	7/20/2020 7:57 PM	File folder	
vioscsi	7/20/2020 7:57 PM	File folder	
vioserial	7/20/2020 7:57 PM	File folder	
viostor	7/20/2020 7:57 PM	File folder	
virtio-win_license	7/20/2020 7:57 PM	Text Document	2 KB
virtio-win-gt-x64	7/20/2020 8:04 PM	Windows Installer ...	5,769 KB
virtio-win-gt-x86	7/20/2020 8:04 PM	Windows Installer ...	4,818 KB
virtio-win-guest-tools	7/20/2020 8:04 PM	Application	25,173 KB

Fig. 3: VirtIO installers

### Binary and config

In the virtual machine, create a *hns-actionmanager* folder at the location *C:\*. Put the stub file in this folder, resulting in the path

*C:\hns-actionmanager\agent-stub\_windows-{arch}*.

Where {arch} is the architecture type (ex: amd64).

Check that everything runs well by executing *C:\hns-actionmanager\agent-stub\_windows-{arch}* in an administrator command prompt. After a few seconds, the stub should be sending heartbeats.

Now that you are sure it works, run the stub executable as administrator to start it. The agent will automatically be registered at each machine startup from now.

## Troubleshooting

If the stub does not appear to be running, check:

- If the serial port box is checked in hyneview
- If the virtIO drivers are installed (see [VirtIO setup](#))

## 3. Usage

### 3.1 Concepts


For the Action Manager, an **action** is any executable with customizable inputs. These actions can be made for Windows or Linux OSes. They are a way to easily instrument a machine and even schedule actions ahead of time with the Scheduler.

When an **action** is started on a virtual machine, it creates a **run**. It is an **action** that is being executed, or has been executed.

Finally, we call **live agent** an agent we can reach from the Action Manager's webui.

### 3.2 Basic features

#### 3.2.1 List topologies

The topologies listing is the main page of the site. You can access it by clicking on the  button. This page displays topologies, whether they are loaded or not, and information regarding them (live entities inside and total number of entities). If you have many topologies, it is possible to search through them by typing their name in the search bar on the left.

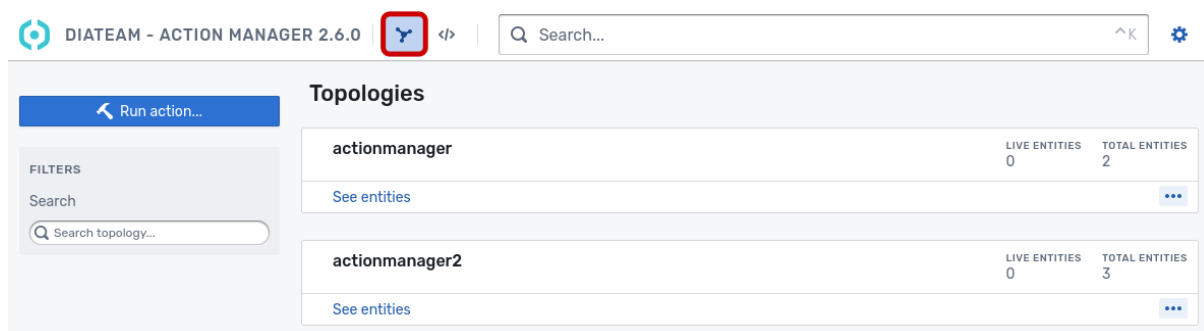


Fig. 1: Topologies listing page

#### 3.2.2 List agents

You can access the agents listing by clicking on the **See entities** button of a specific topology.



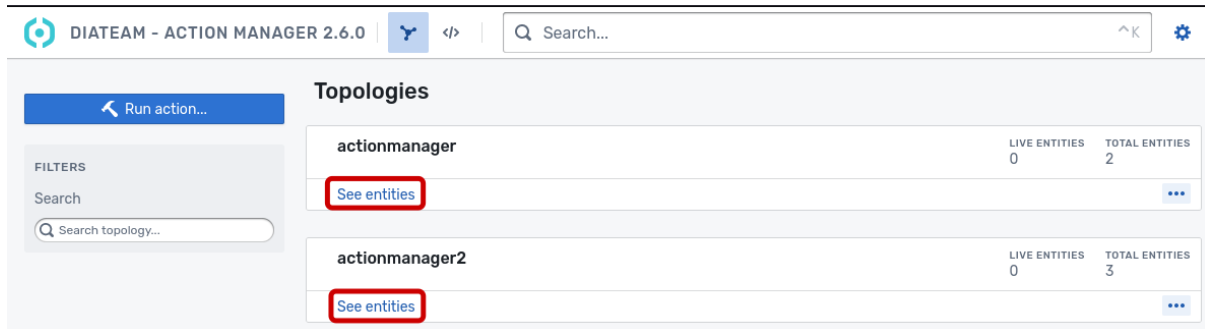


Fig. 2: **See entities** button

This page displays entities, whether they're live or not, and information regarding them (total number of runs, topologies, etc). You can display only live entities if you need to and find an entity by typing its name with the menu on the left side.

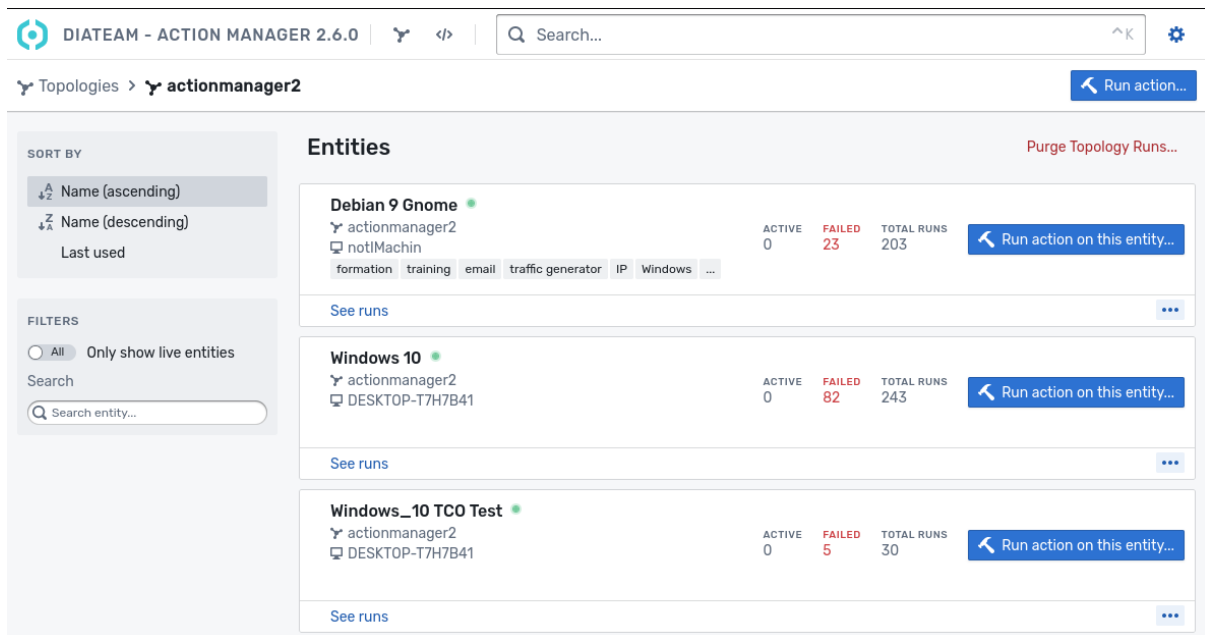


Fig. 3: Entities listing page

### 3.2.3 List runs

To see runs from a specific entity, click on **see runs** under the entity.

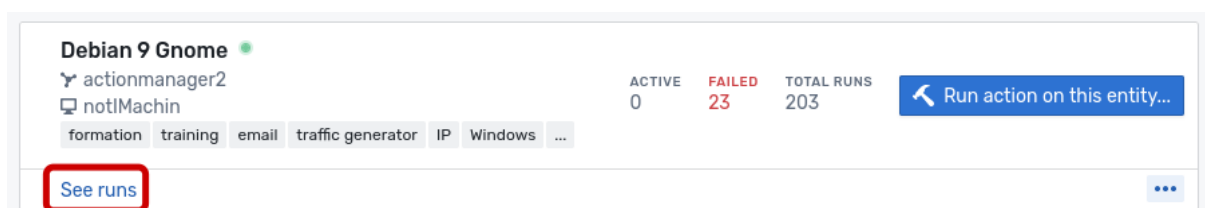
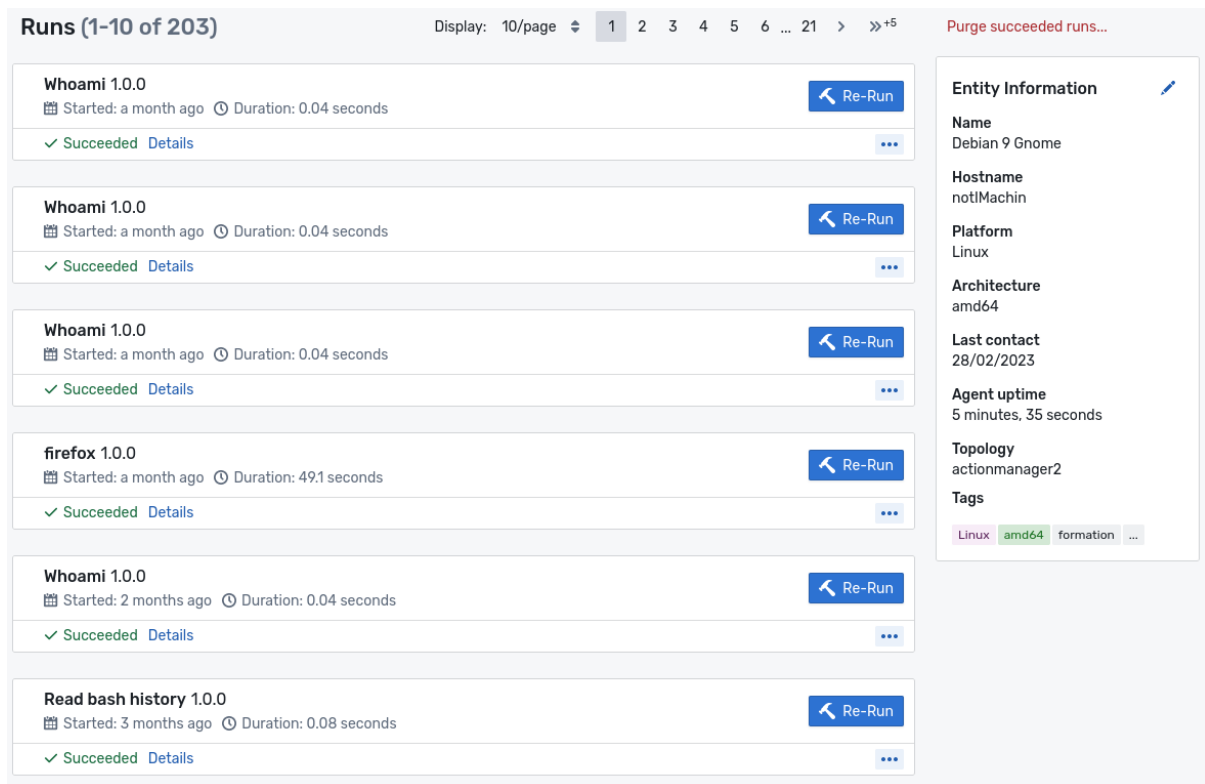


Fig. 4: **See runs** button

Runs appear in chronological order and can be restarted by clicking **Re-run**.



**Runs (1-10 of 203)** Display: 10/page 1 2 3 4 5 6 ... 21 > >>+5 Purge succeeded runs...

- Whoami 1.0.0**  
Started: a month ago Duration: 0.04 seconds  
Succeeded Details Re-Run
- Whoami 1.0.0**  
Started: a month ago Duration: 0.04 seconds  
Succeeded Details Re-Run
- Whoami 1.0.0**  
Started: a month ago Duration: 0.04 seconds  
Succeeded Details Re-Run
- firefox 1.0.0**  
Started: a month ago Duration: 49.1 seconds  
Succeeded Details Re-Run
- Whoami 1.0.0**  
Started: 2 months ago Duration: 0.04 seconds  
Succeeded Details Re-Run
- Read bash history 1.0.0**  
Started: 3 months ago Duration: 0.08 seconds  
Succeeded Details Re-Run

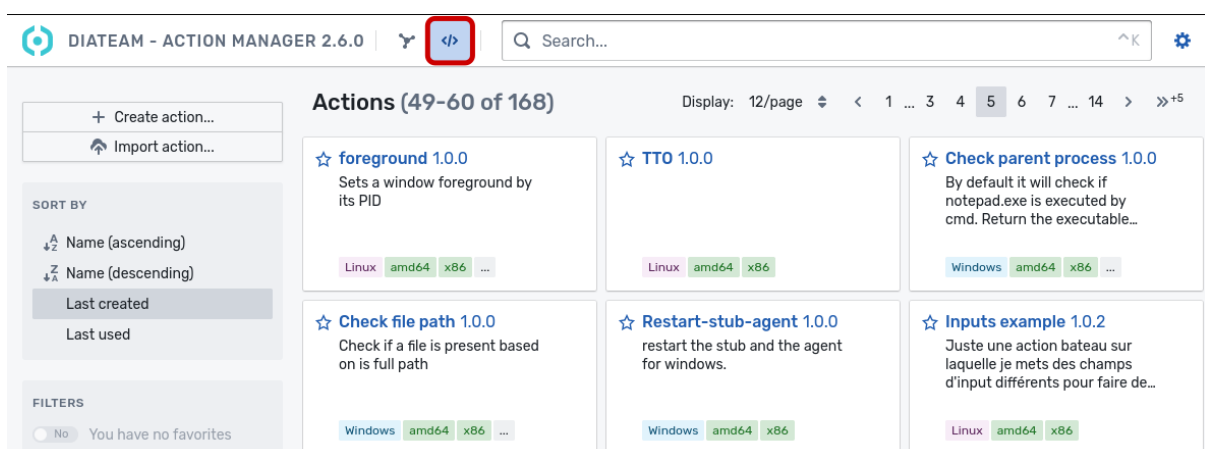
**Entity Information**

- Name:** Debian 9 GNOME
- Hostname:** notIMachin
- Platform:** Linux
- Architecture:** amd64
- Last contact:** 28/02/2023
- Agent uptime:** 5 minutes, 35 seconds
- Topology:** actionmanager2
- Tags:** Linux, amd64, formation, ...

Fig. 5: Runs list of a specific entity

### 3.2.4 List actions

The **Actions** (</>) page lists all the actions uploaded on the Action Manager. You can filter them as in the entities' page using the menu on the left side.



DIATEAM - ACTION MANAGER 2.6.0 </> Search...

**Actions (49-60 of 168)** Display: 12/page < 1 ... 3 4 5 6 7 ... 14 > >>+5

- foreground 1.0.0**  
Sets a window foreground by its PID  
Linux amd64 x86 ...
- TTO 1.0.0**  
Linux amd64 x86
- Check parent process 1.0.0**  
By default it will check if notepad.exe is executed by cmd. Return the executable...  
Windows amd64 x86 ...
- Check file path 1.0.0**  
Check if a file is present based on its full path  
Windows amd64 x86 ...
- Restart-stub-agent 1.0.0**  
restart the stub and the agent for windows.  
Windows amd64 x86
- Inputs example 1.0.2**  
Juste une action bateau sur laquelle je mets des champs d'input différents pour faire de...  
Linux amd64 x86

**CREATE ACTION**

- + Create action...
- Import action...

**SORT BY**

- Name (ascending)
- Name (descending)
- Last created
- Last used

**FILTERS**

- No You have no favorites

Fig. 6: Actions list page

### 3.2.5 Actions versioning

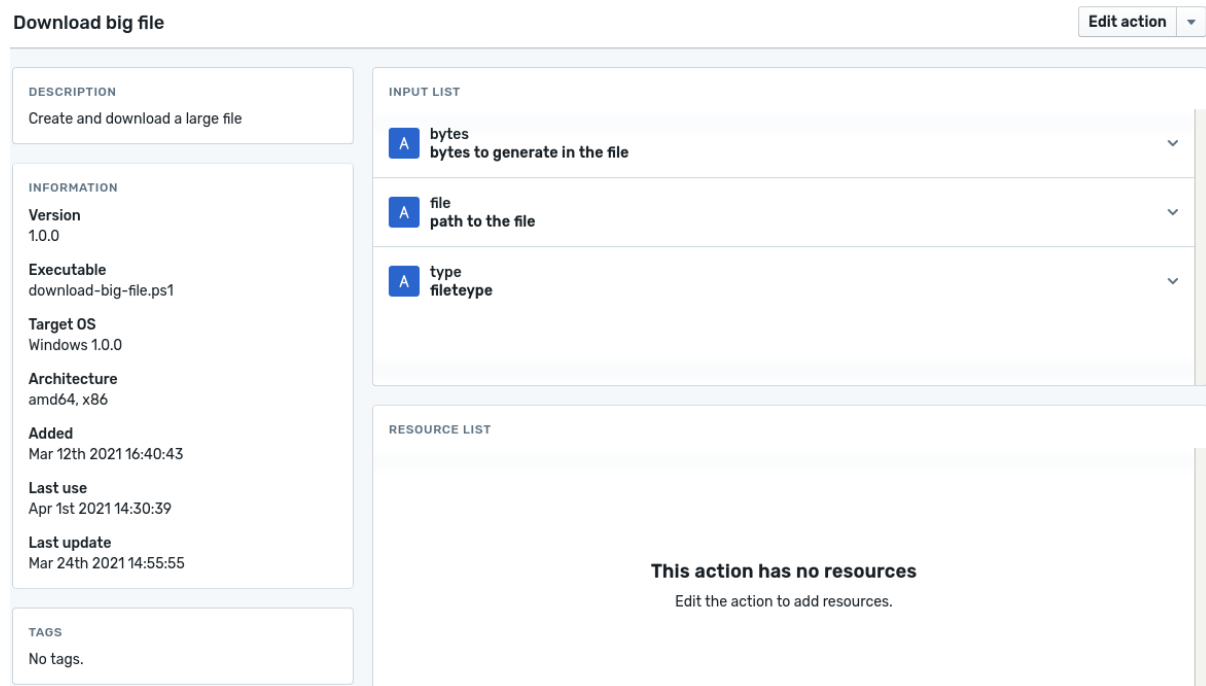
Each action is given a version at its creation. You can set it by yourself, the default value is 1.0.0. It allows you to have actions with the same name but different executable, depending on your needs.

It is also a way to differentiate your actions between those imported and those you created.

### 3.2.6 Get action details

You can edit, download, duplicate or delete the action using the menu in the bottom right corner.

Clicking on an action title will redirect you to a detailed view of the action with the list of its inputs, resources and details about its use.



**Download big file** Edit action ▾

**DESCRIPTION**  
Create and download a large file

**INFORMATION**

**Version**  
1.0.0

**Executable**  
download-big-file.ps1

**Target OS**  
Windows 1.0.0

**Architecture**  
amd64, x86

**Added**  
Mar 12th 2021 16:40:43

**Last use**  
Apr 1st 2021 14:30:39

**Last update**  
Mar 24th 2021 14:55:55

**TAGS**  
No tags.

**INPUT LIST**

- bytes to generate in the file
- file path to the file
- filetype

**RESOURCE LIST**

**This action has no resources**  
Edit the action to add resources.

Fig. 7: Actions details page

### 3.2.7 Create actions

To create a new action, go to the **Actions** page and click on the **Create action...** button.

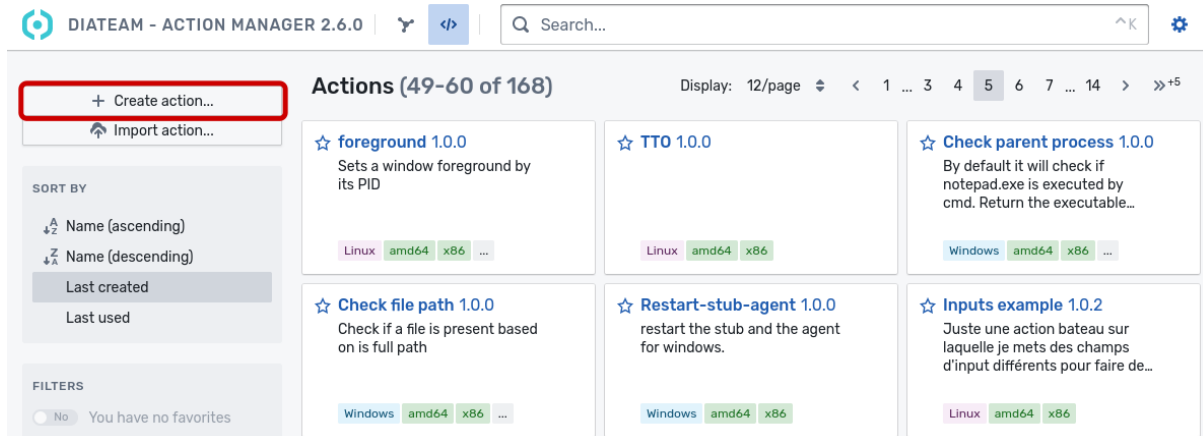
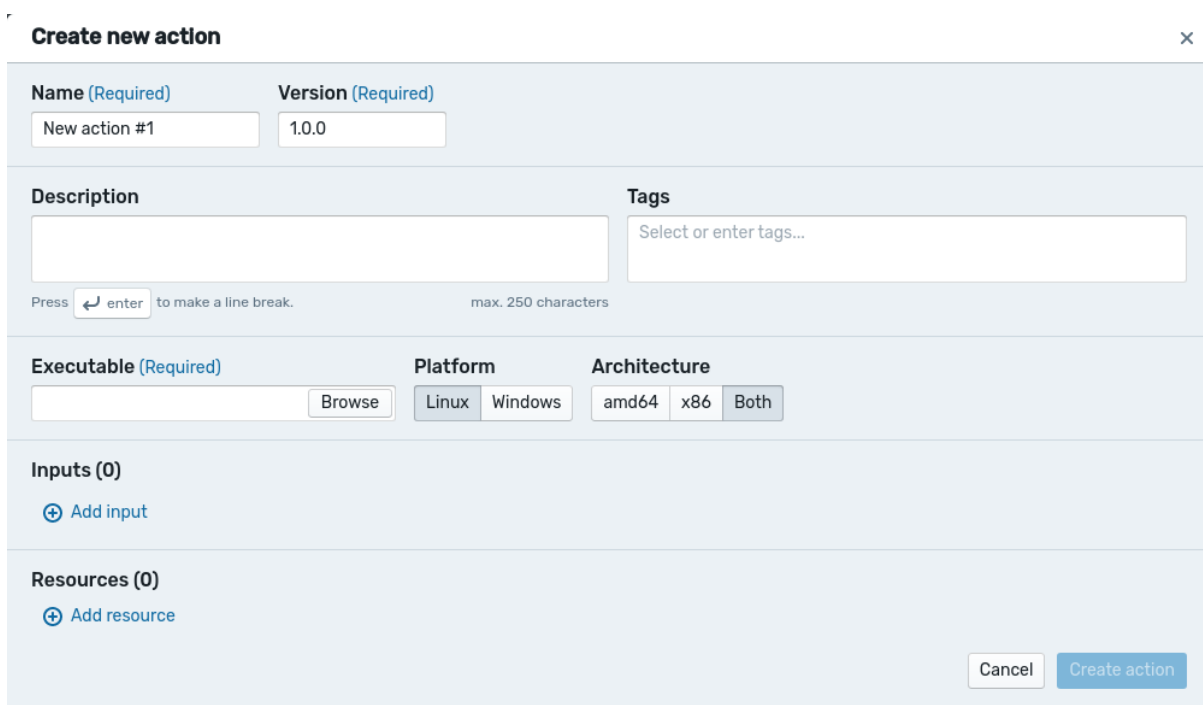


Fig. 8: Create an action

You will then be prompted with the **Create new action** form.



The 'Create new action' form is a modal window with the following fields and options:


- Name (Required):** Input field containing 'New action #1'.
- Version (Required):** Input field containing '1.0.0'.
- Description:** Text area with a placeholder and a note: 'Press  enter to make a line break. max. 250 characters'.
- Tags:** Input field with a placeholder 'Select or enter tags...'.
- Executable (Required):** Input field with a 'Browse' button.
- Platform:** Radio buttons for 'Linux' and 'Windows'.
- Architecture:** Radio buttons for 'amd64', 'x86', and 'Both'.
- Inputs (0):** Section with an 'Add input' button.
- Resources (0):** Section with an 'Add resource' button.
- Buttons:** 'Cancel' and 'Create action' buttons at the bottom right.

Fig. 9: Create action popup

Fill in the fields needed for your action:

- Name: the action name
- Version: the action version (see [Actions versioning](#))
- Description (*optional*): details about the action, used in research
- Tags (*optional*): keywords to describe your action, used to filter actions when listing them
- Executable: the script that will be run on the entity (.ps1, .bat or .sh)

- Platform: the OS on which the script is made to run
- Architecture: the architecture of the agent's OS
- Inputs (*optional*):
  - Name: name of input
  - Type: data type
  - Description: details about the input
  - Default: value used if the field is not filled in
  - Example: value suggested
- Resources (*optional*): one or more files packaged with the action

---

**Note:** The purpose of resources is to have one or more files packaged directly with the action. Meaning they are saved between runs and it is not needed to reupload them each time, unlike inputs.

---

### 3.2.8 Import actions

If you downloaded an action (.am or .zip format). You can import it to the Action Manager. In the **Actions** page, click on **Import action....** You will be prompted with a popup to add your action. You can either choose to drag and drop it or find it on your machine with the button **Select file**.

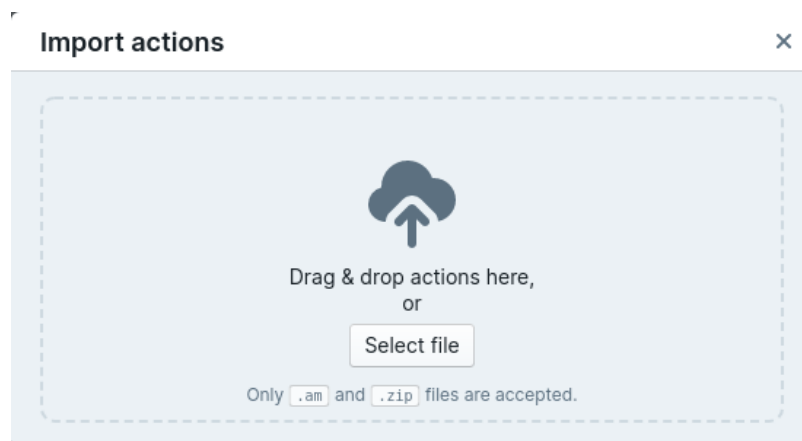


Fig. 10: Popup for action import

### 3.2.9 Download actions

If you need to share an action to another instance of the Action Manager or wish to make a backup of your actions, you can download them.

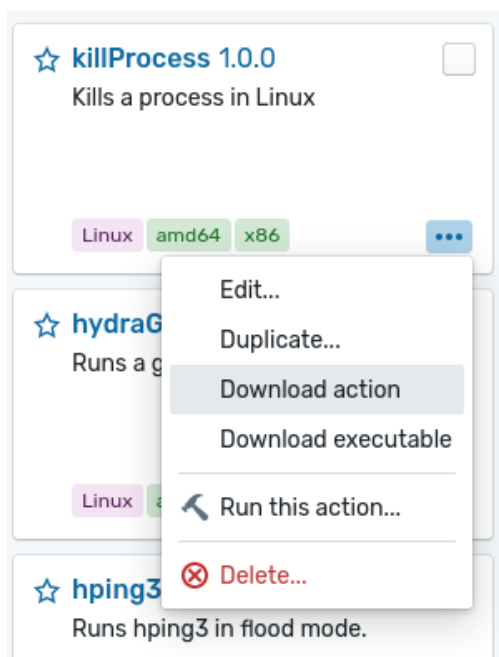


Fig. 11: Download an action

## Import strategies

There are 4 options to choose from when importing actions to handle conflicts between existing actions and actions being imported.

- **Error:** the default, returns an error if there is a conflict
- **Create or replace:** replaces the existing action with the action imported
- **Ensure exists:** keeps the existing action, discarding the imported action
- **Increase version:** keeps the existing action to its version, increments the version of the imported action (see [Actions versioning](#))

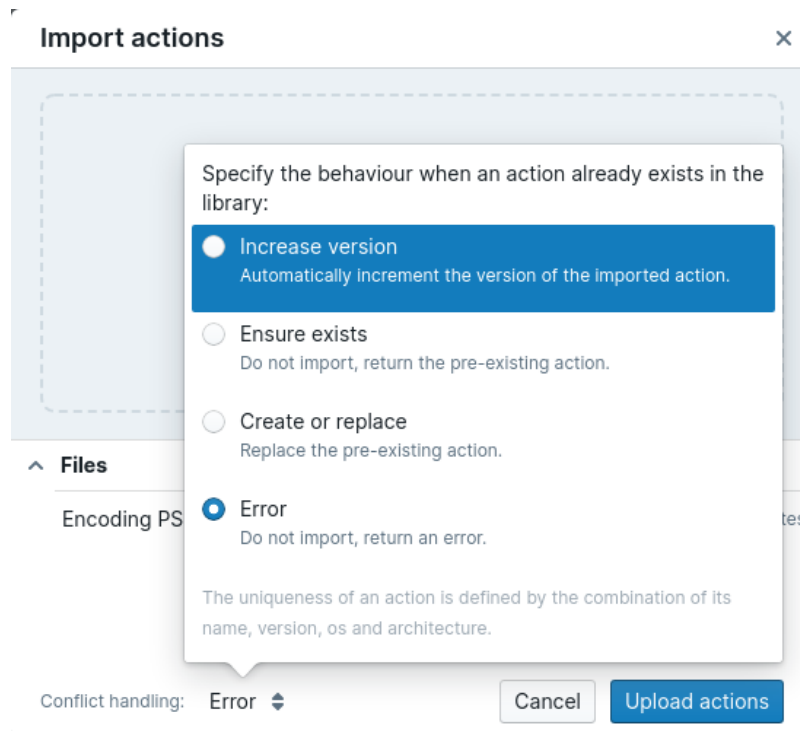


Fig. 12: **Conflict handling** dialog

### 3.2.10 Run actions

There are several ways to run an action:

- On the **Entities** page (when you select a topology)
  - with the button **Run action**
  - with the button **Run action on this entity...** on each entity
- On the **Actions** page with the button **...** on each action
- On the **Action page**, when you select an action, click on the arrow right of the **Edit action** button and **Run this action....**
- On the runs listing, every action has a button **Re-Run**

By using any of these ways you should be prompted with the action **Launcher**.

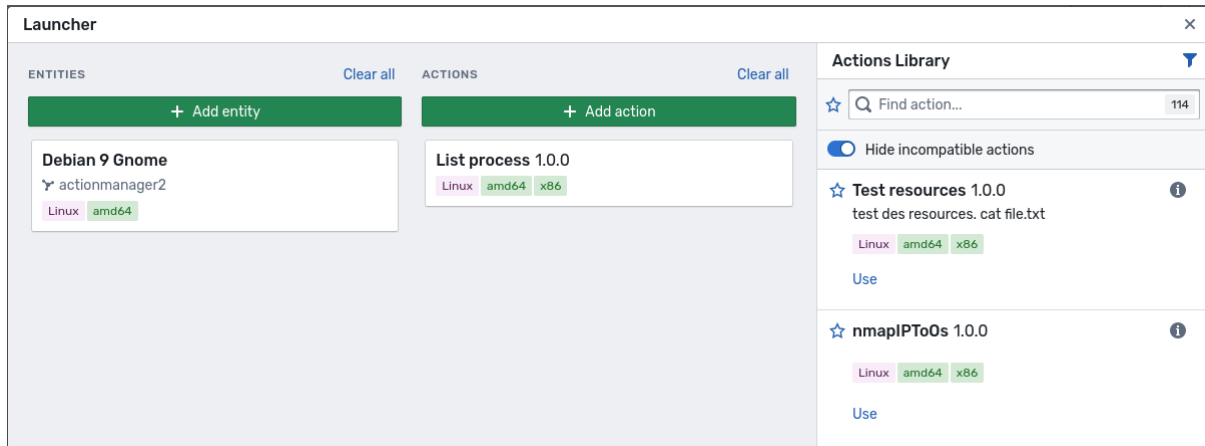


Fig. 13: Action launcher

The action **Launcher** is divided in 3 columns:

- The list of entities selected for a run
- The list of actions that will be performed on these entities
- The menu with which you can choose your entities, actions, and inputs

If you want the menu on the right to display the list of entities, click on **Add entity**. If you want the list of actions, click on **Add action**. Finally, if you want the inputs for a specific action click on the action tile.

---

**Note:** Actions with available inputs appear with a ⚙️ at the bottom right of their tile.

---

### 3.2.11 Purge runs

When you have run many actions, the run page and the quick-view of the active, failed and total runs on an entity can become cluttered. There are two buttons to purge the runs of a whole topology. One of them is located in the **Topologies** page, in the options of any topology. The other one is located on the **Entities** page of any topology, above the list of all entities.

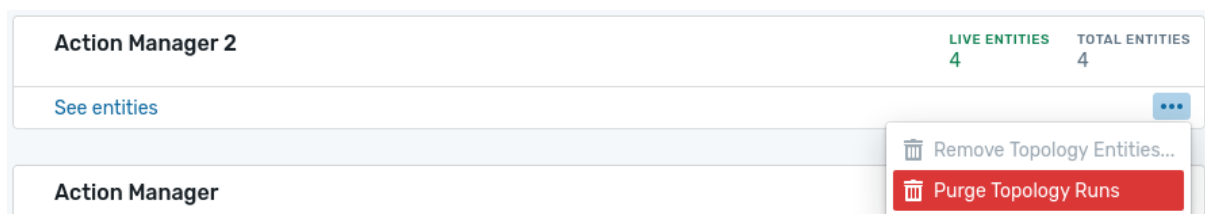


Fig. 14: Purge all runs of a topology, **Topologies** page



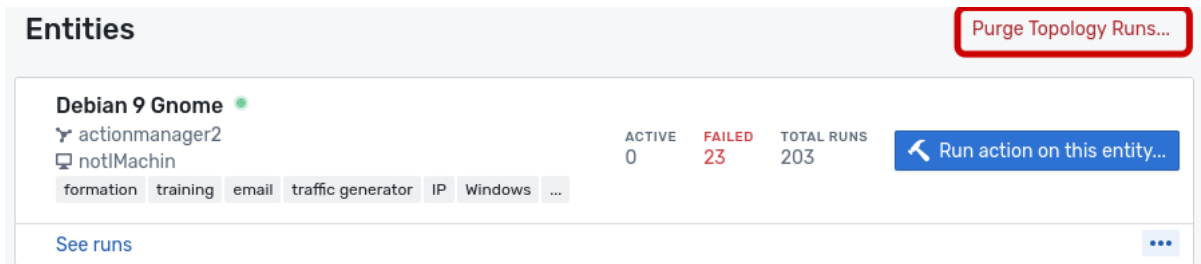


Fig. 15: Purge all runs of a topology, **Entities** page

By going to the **Runs** page of an entity, there is also a way to **Clean succeeded runs...**, which will leave only the failed or running runs on display. Lastly, it is possible to delete each run one by one, by clicking on **Delete** in the options of every run on this page.

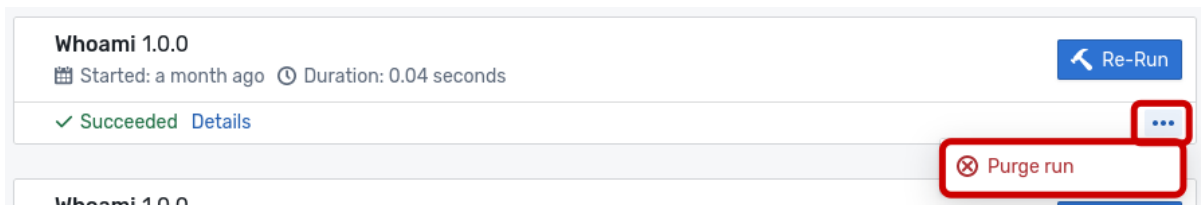


Fig. 16: Purge a specific run

### 3.2.12 Remove entities

If there are entities that you no longer use, it is possible to remove them from the Action Manager's listings. You can go to the **Entities** page and click on the options button, then select **Remove entity...**

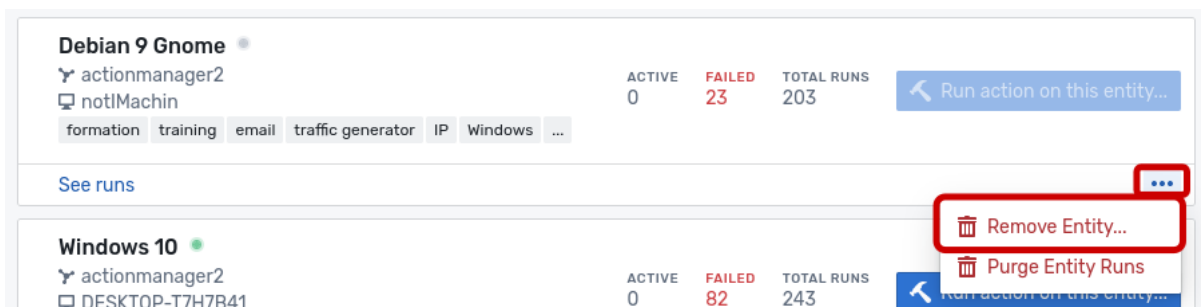


Fig. 17: Removing a single entity

**Note:** The entity must be stopped in hynesim to remove it from the AM. If it is restarted, it will be displayed again within a few seconds.

It is possible to remove all the entities from a topology, too. In the **Topologies** page, click on the options button of a topology and select **Remove Topology Entities...**

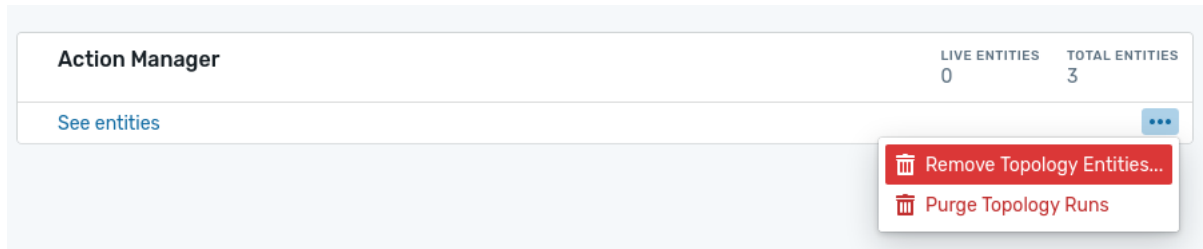


Fig. 18: Removing several entities

**Note:** A topology will disappear from the **Topologies** page if all of its entities are removed. It will be displayed again if any of its entity is restarted.

## 3.3 Advanced features

### 3.3.1 Scheduler

The scheduler allows you to postpone an action run and have it run at any time you want. It is located in the action **Launcher**, click on the action you want to postpone. In the panel on the right, there is a **Not scheduled** link. Click on it and choose the date and time you want the action to run at.

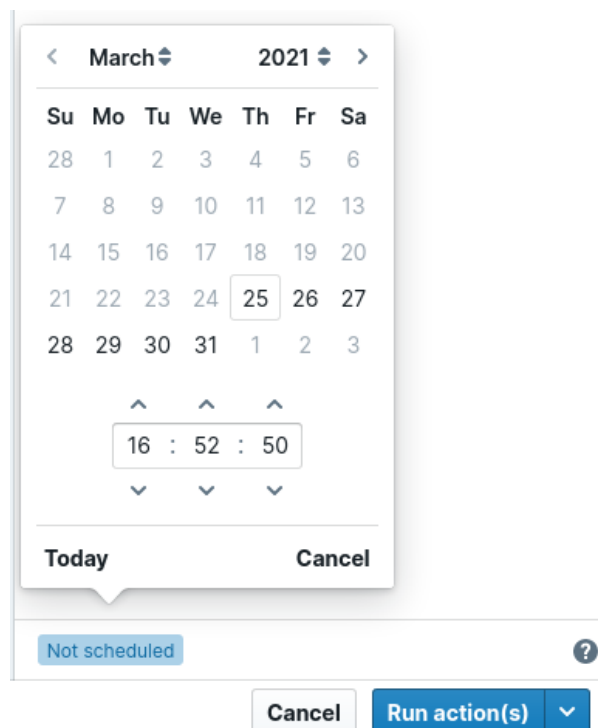


Fig. 19: Schedule an action

### 3.3.2 Action inputs

Upon action creation, you can specify inputs. These inputs can either be a file if you need to compare it to a file in the VM for instance, or text which can be a flag to a shell command.

Let's create a simple action, which returns the hardware address of a specific network interface. First, create a .sh script on your host and paste this code in it:

```
#!/bin/bash
cat /sys/class/net/${1}/address
```

This script could be run directly in a linux shell by typing:

```
./macAddr.sh eth0
```

It would output the physical address of the argument, the interface **eth0**.

Now, we want to have this script as an action in the Action Manager, create it as such:

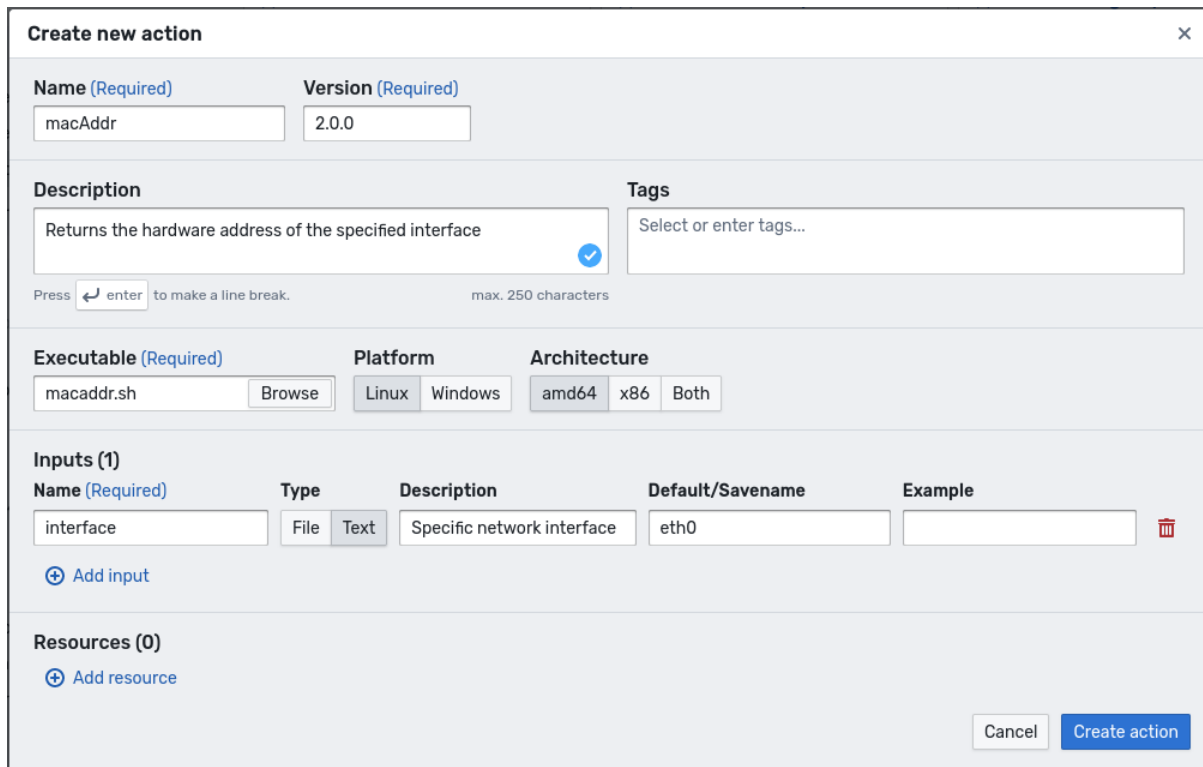


Fig. 20: Creating macAddr action

The input “Interface” will result in a form in the action launcher, in which you can specify any interface. This form is accessible by adding the action to the list of actions to run and click on its card in the list.

This input is going to be the argument of the script.

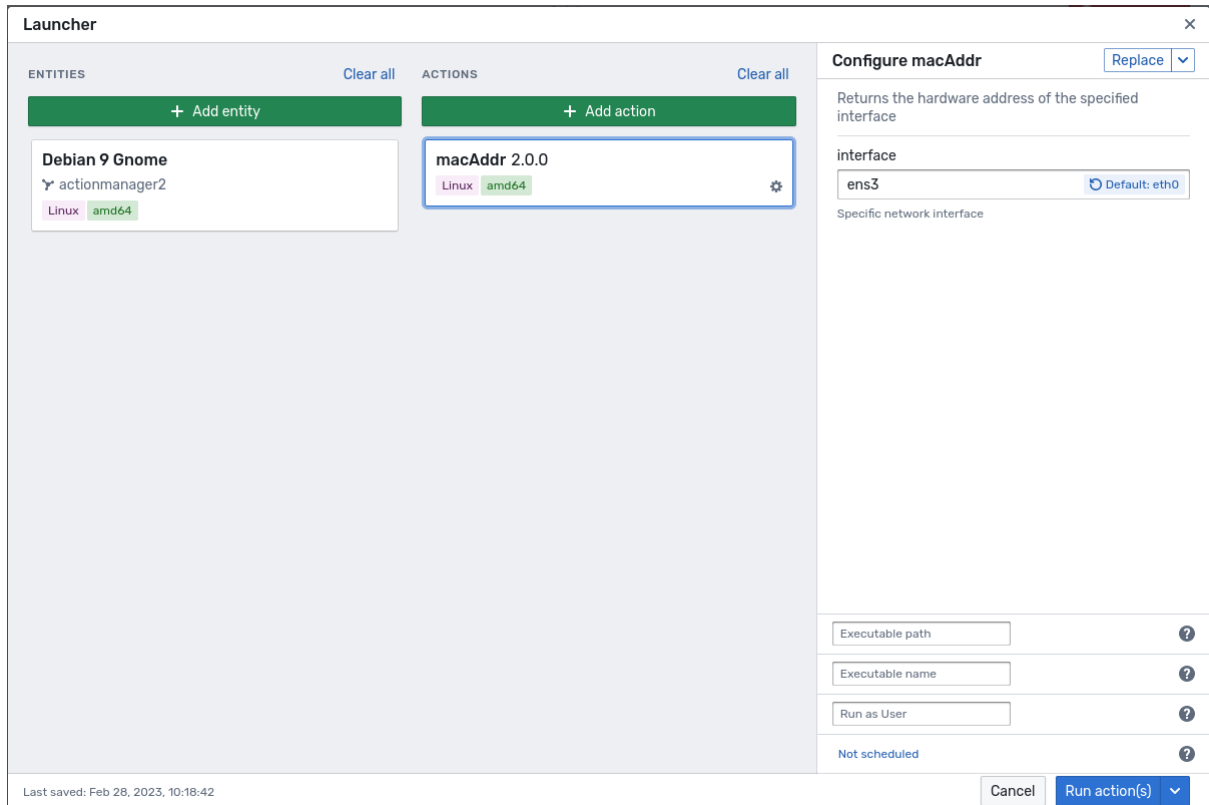


Fig. 21: Running macAddr action

**Note:** In this example, we chose “eth0” as a default but it is up to you to choose the default you want or even not put any default at all.

Here is the result, it will obviously vary depending on the VM’s hardware addresses.

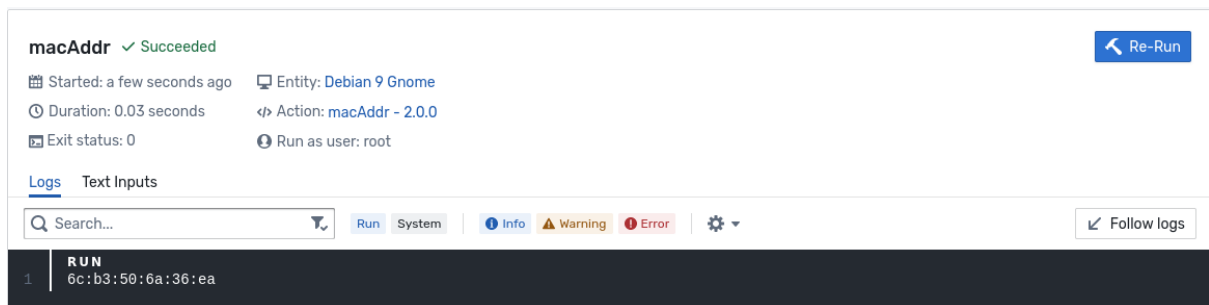
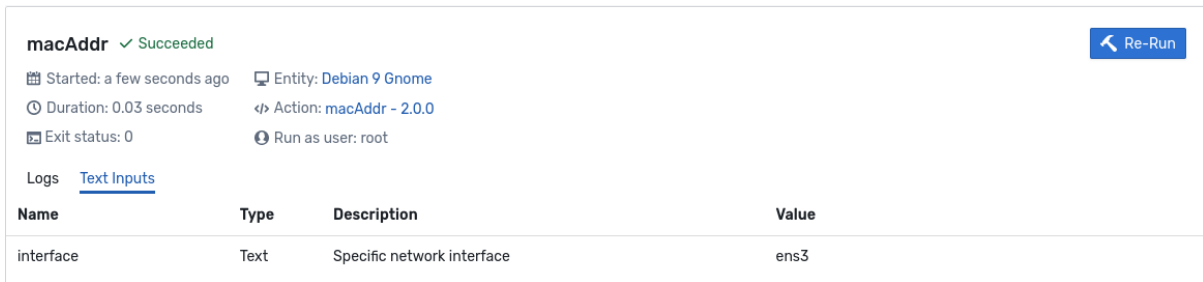


Fig. 22: Logs of macAddr run

If you need to, you can check the provided input of any action at any time in the “Text Inputs” tab. It should list every text input with its name, description and value. When there are file inputs, they are located in a “File Inputs” tab.



**macAddr** ✓ Succeeded Re-Run

Started: a few seconds ago Entity: Debian 9 Gnome  
Duration: 0.03 seconds Action: macAddr - 2.0.0  
Exit status: 0 Run as user: root

Logs Text Inputs

Name	Type	Description	Value
interface	Text	Specific network interface	ens3

Fig. 23: Text inputs tab

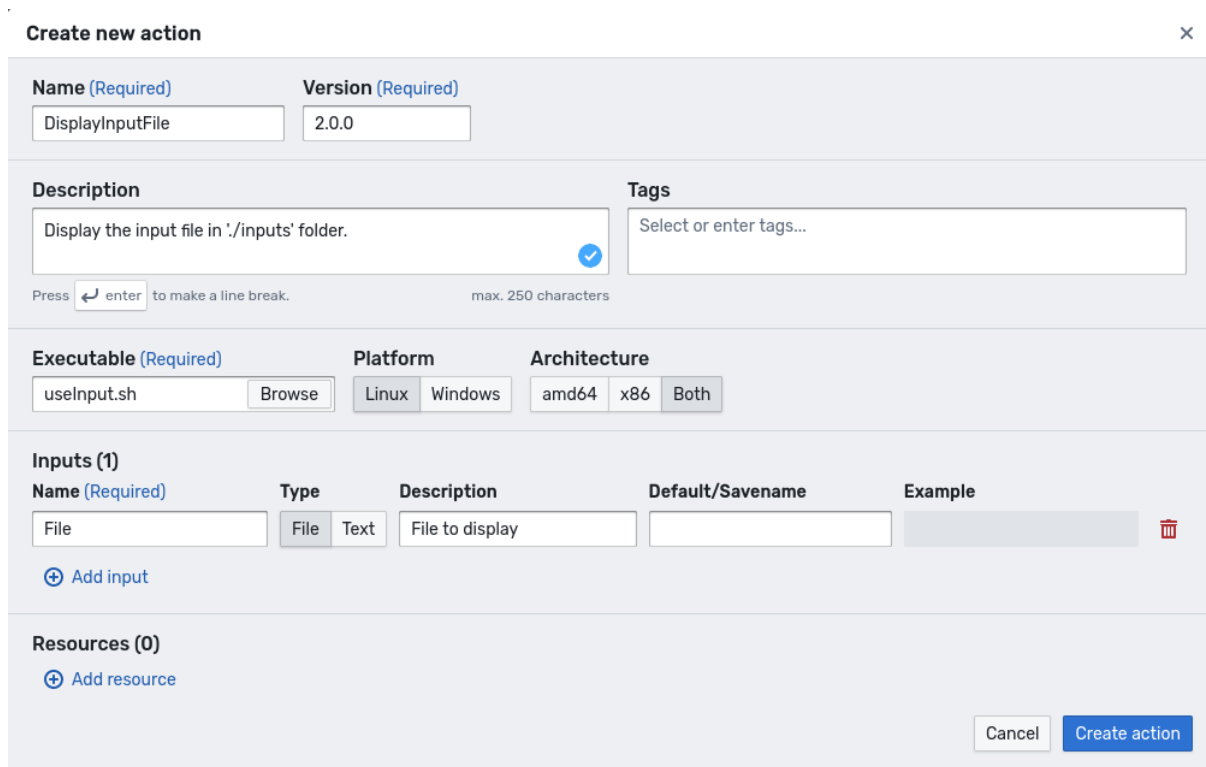
It works the same way with input files. The only thing that changes at the action creation is the **Type** selector, which you have to switch to **File**. Files used as inputs will be located in the “./inputs” folder.

Here is a script to use file inputs as an example:

```
#!/bin/bash
cat ./inputs/file.txt
```

It will display in the Action Manager logs the content of the file “file.txt”, located in the folder “./inputs”.

Now, create an action with this script:



**Create new action** [x]

**Name (Required)**  **Version (Required)**

**Description**  **Tags**

Press  enter to make a line break. max. 250 characters

**Executable (Required)**   **Platform**   **Architecture**

**Inputs (1)**

Name (Required)	Type	Description	Default/Savename	Example
<input type="text" value="File"/>	<input type="button" value="File"/> <input type="button" value="Text"/>	<input type="text" value="File to display"/>	<input type="text"/>	<input type="text"/>

**Resources (0)**

Fig. 24: Create action with input file

Run the action, to select the file to use as input, there is a button to browse your filesystem and find it easily.

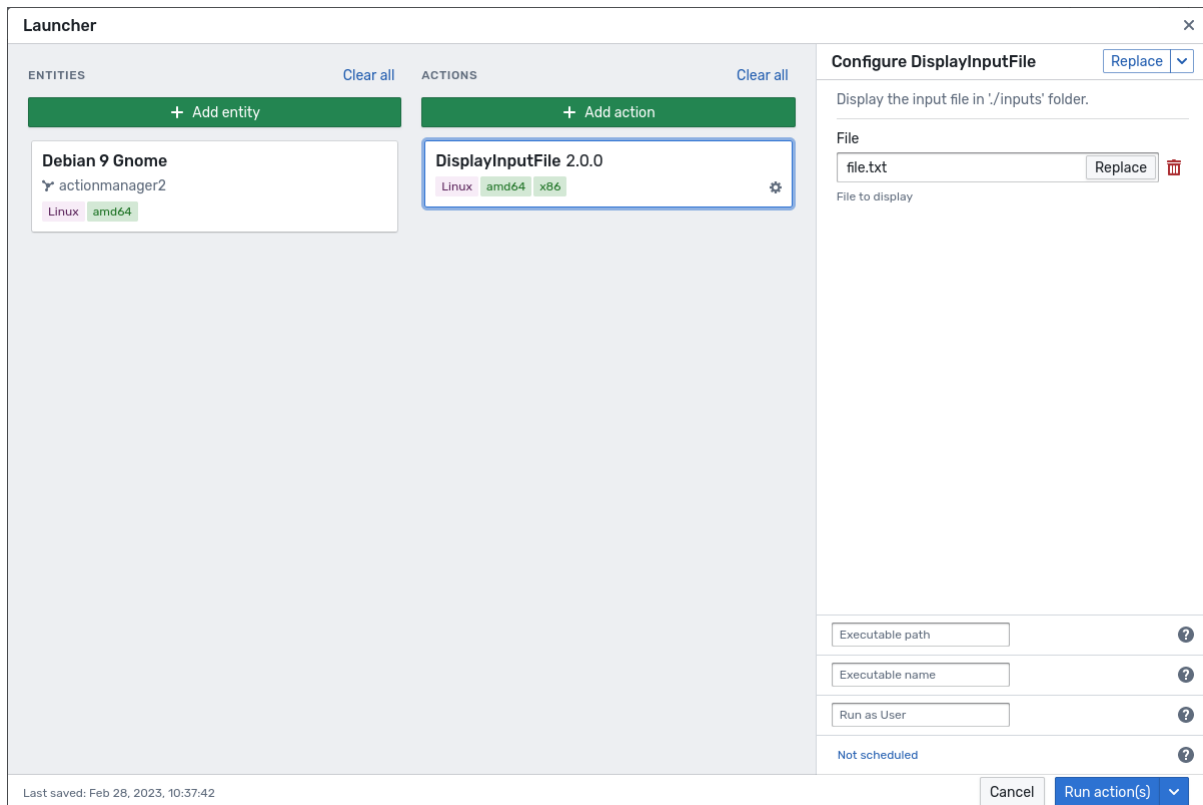


Fig. 25: Run action with input file

The Action Manager should display the content of the file you used as input.

### 3.3.3 Action resources

If you need to use a file every time you run an action, it can be tedious to put it in input each and every run. For instance, if you need to install several times a software with source files, it is really practical to have them always packaged in the action. That is why there are resources. They are files packaged with an action at its creation time.

Let's use them in an example. Here is the script we use:

```
#!/bin/bash
cat "resources/file.txt"
```

We upload a file.txt file to be displayed in the logs. It could be any other file format.

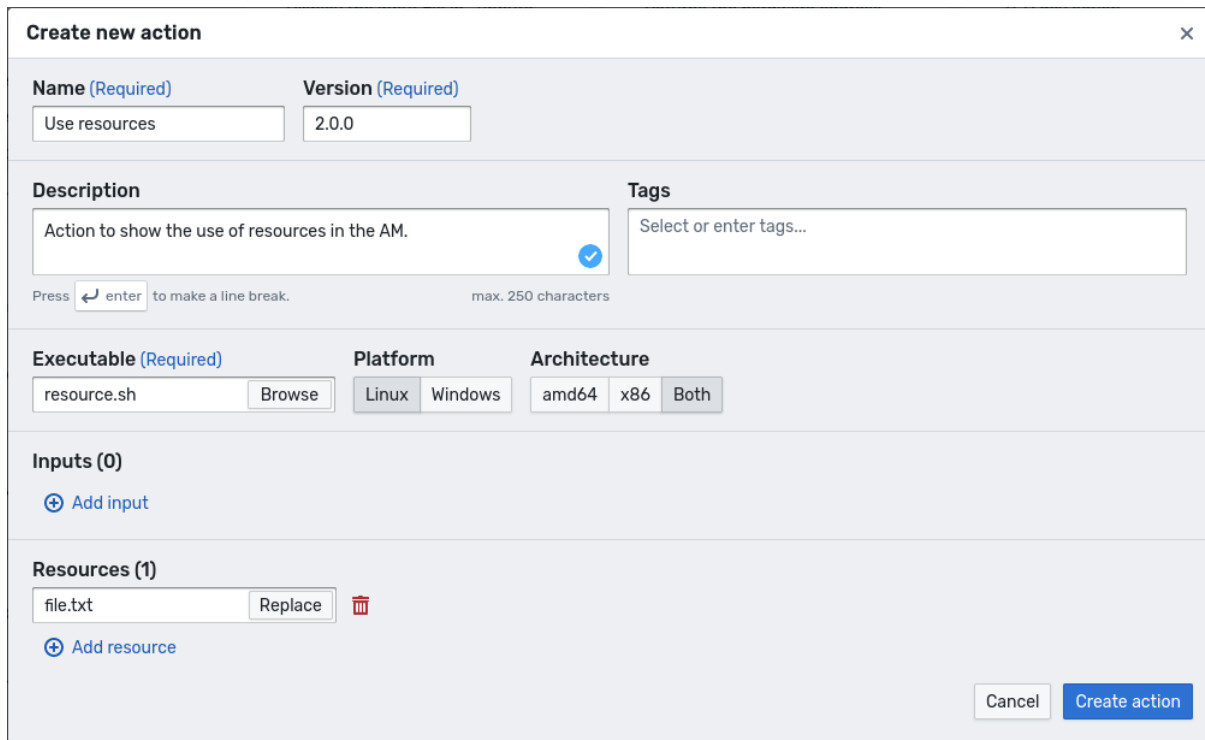


Fig. 26: Create resource using action

This script simply displays the content of the file used as resource, here is the output:



Fig. 27: Resource display result

---

**Note:** Resources are at the same location on Linux or Windows, in the “./resources” folder.

---

### 3.3.4 Action outputs

If you need to store an action’s output data on a file for easier handling, there is the `./outputs` folder. Files stored in this folder during a run are sent to the host.

Here is an example script:

```
#!/bin/bash
echo 'This is going to the output files' > ./outputs/file.txt
```

Now, create the action:

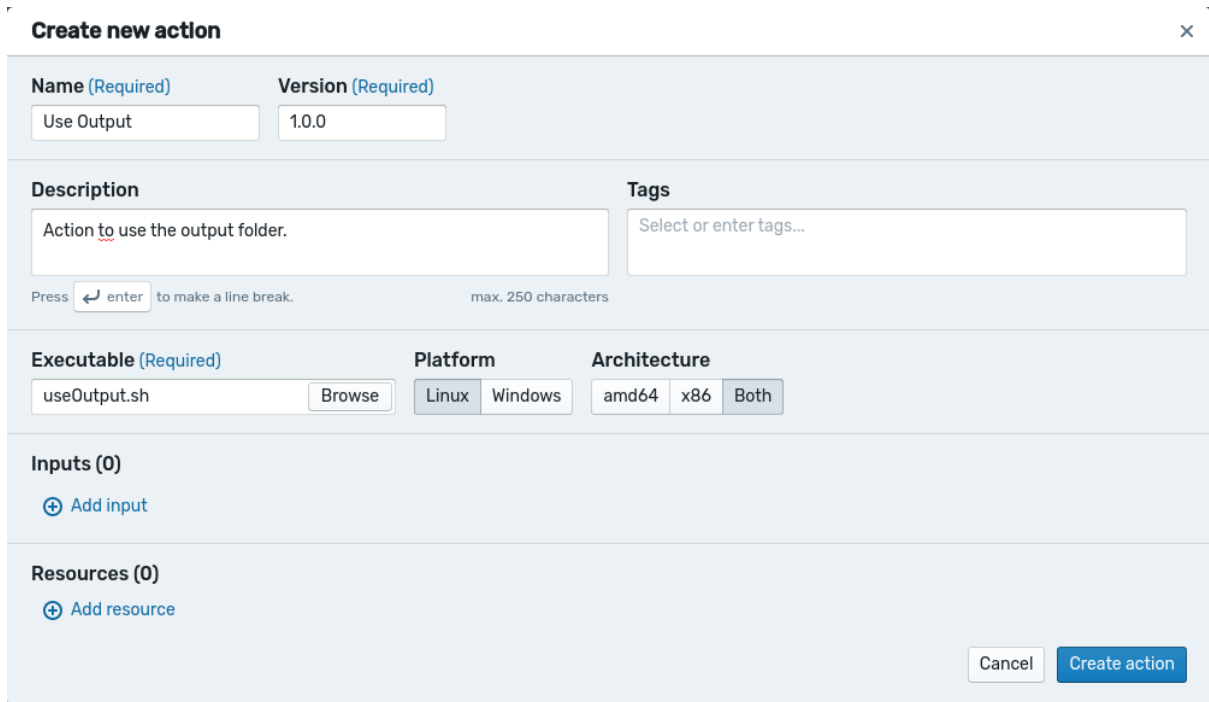


Fig. 28: Create action using output

Run the action as a basic action, there is no input to handle.

Once the action has run, you can find the output files in the **Outputs** tab on the run page.

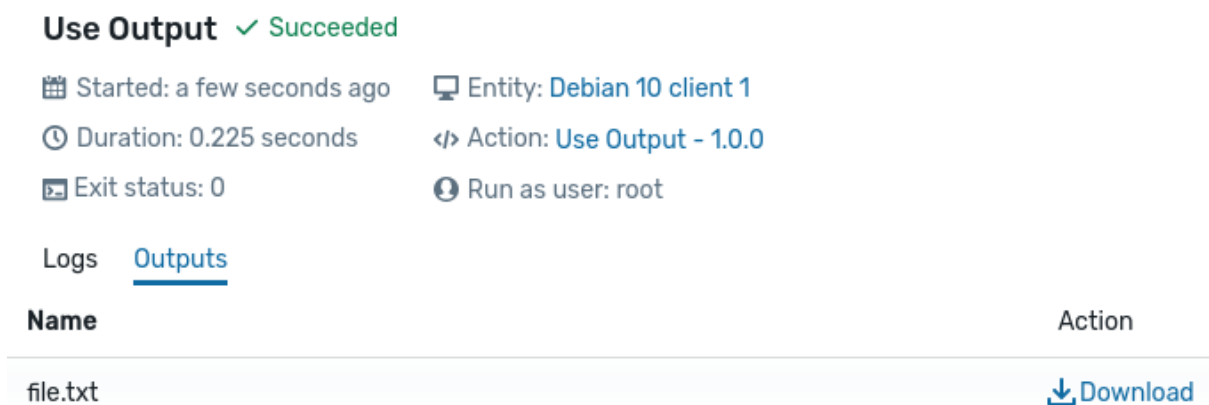


Fig. 29: **Outputs** tab on run page

There you can directly download your output files.



### 3.3.5 User actions

By default the actions are run as administrator on the machines. The administrator is the nt authority on Windows and root on Linux. It does not let us pop graphical applications or do user-specific actions.

This is why we can impersonate a user to run an action. It is done by specifying its username in the action launcher, when we have an action selected. If you don't want to run the action as a user, leave the field empty.

Here are two example scripts for Linux and Windows, they both pop a calculator:

Linux:

```
#!/bin/sh
export DISPLAY=:0 # This line is to let Linux know where to instantiate the app
gnome-calculator
```

Windows (powershell):

```
C:\Windows\System32\calc.exe
```

Now, specify the user that is going to run the action, “hns” in our case:

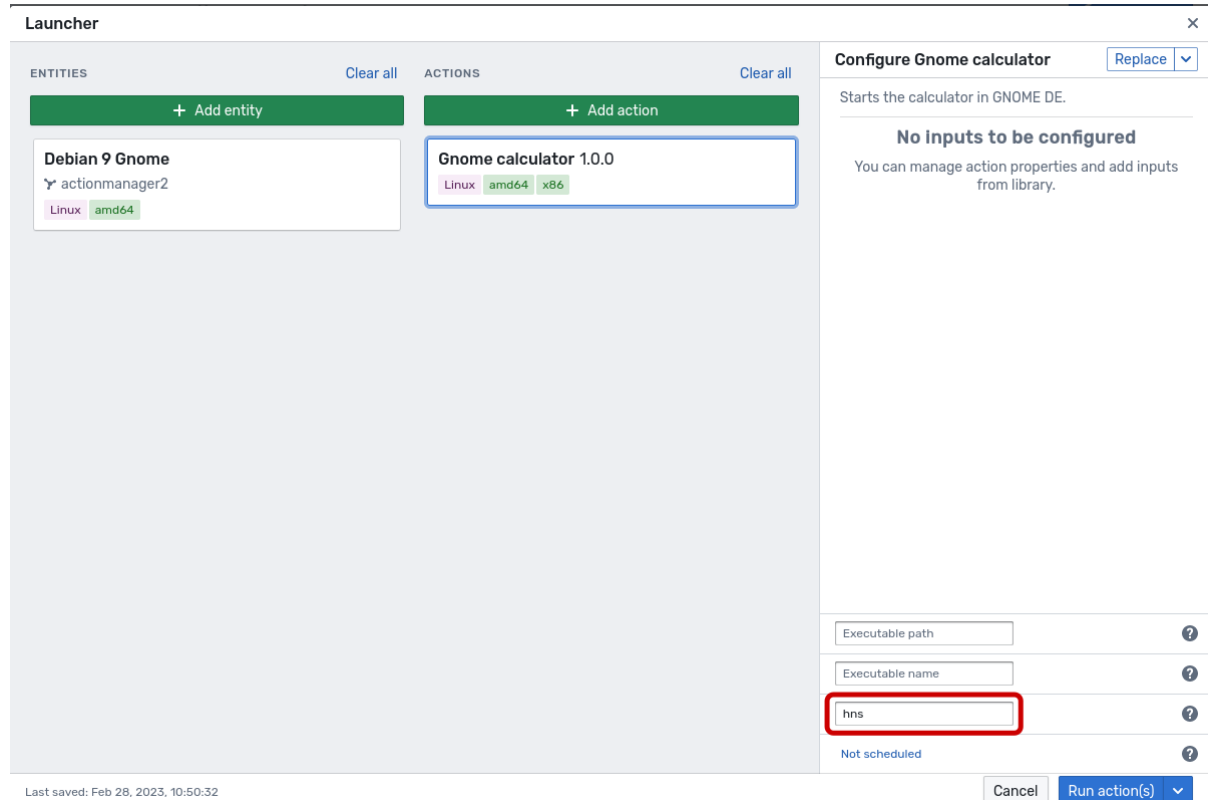


Fig. 30: Run action as user “hns”

**Note:** For Windows actions, it is mandatory to have the specified user logged in his session.

Otherwise it will not work.

As a result, you should see a calculator pop open on the screen of the specified user.

### 3.3.6 Executable name

By default the action will be named like **exec**, but you have the possibility to rename the action.

You just have to specify the name that you want use, and in the VM the file will be renamed:

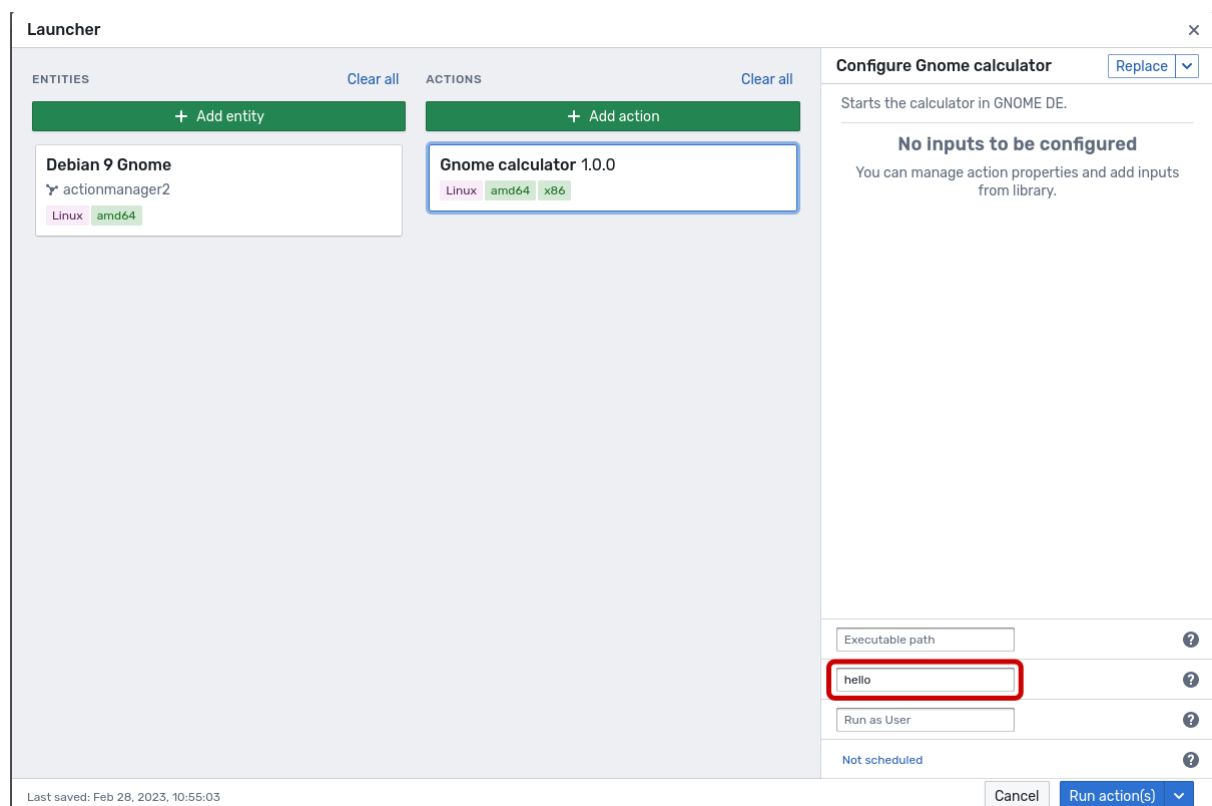


Fig. 31: Change the executable name

**Note:** You can omit the extension, by default the AM use **.ps1** for Windows and in Linux the executable will be run using Bash.

### 3.3.7 Executable path

By default the action will be executed in the temporary directory. You can change the directory that the action will be executed.

You just have to specify the path that you want use, and in the VM the action will be upload inside:

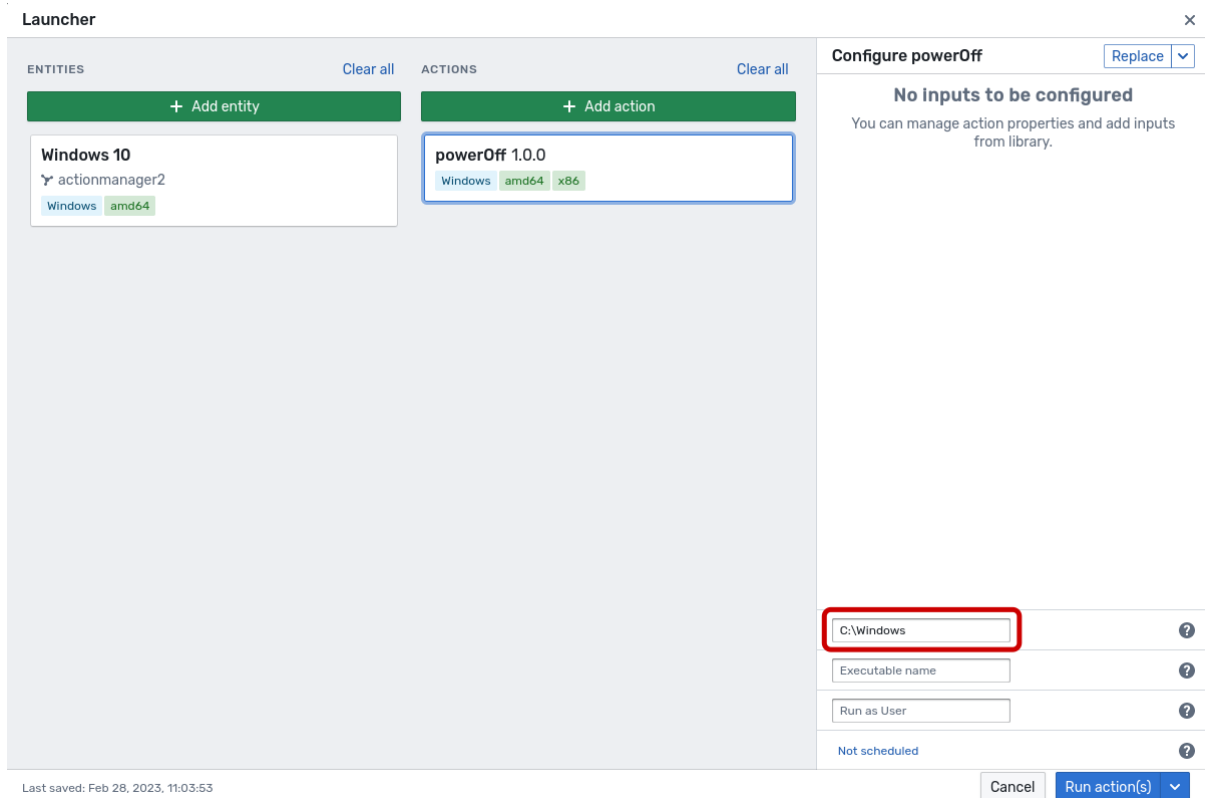


Fig. 32: Change the executable name

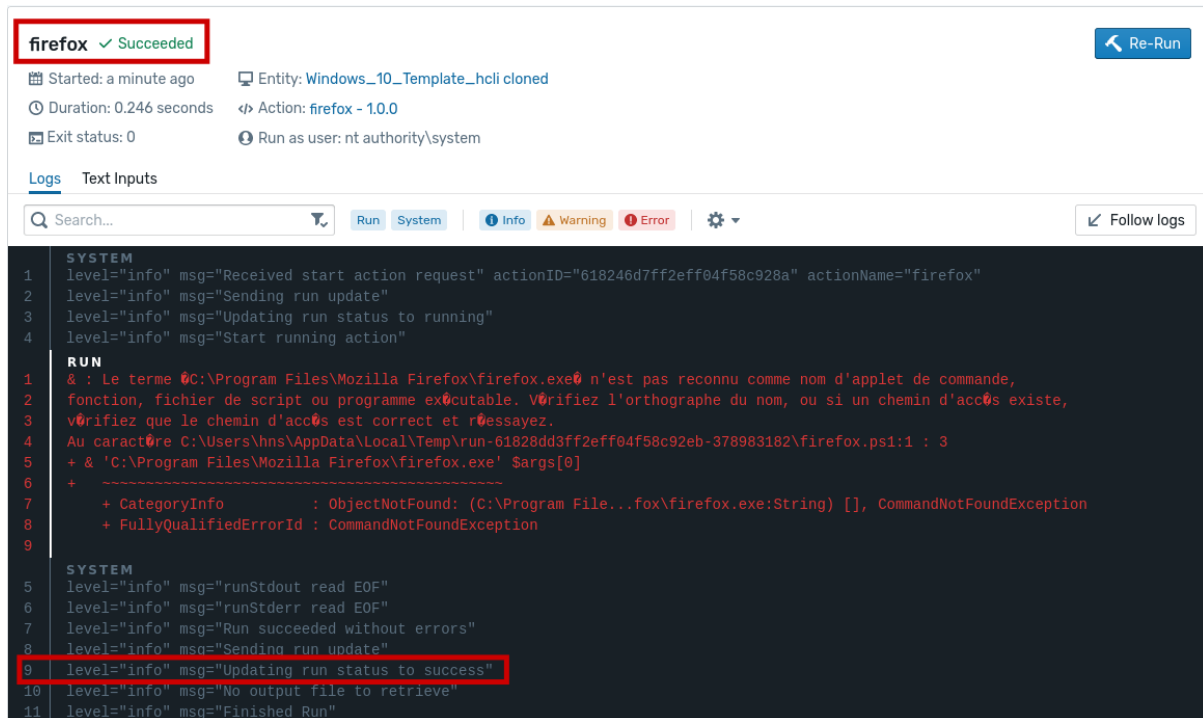
## 3.4 Error management

When you create a script using Windows PowerShell, batch or Linux bash, you should test your program to define the return error cases.

For example:

```
& "C:\Program Files\Mozilla Firefox\firefox.exe"
```

This action will return **Success** but the created process is failing because of an external reason. What is actually happening is that we get the status of the process starting the action itself. As it started the action successfully, the returned value is **Success**, even if the action itself failed, this use case happens when you don't handle errors in your script.



**firefox** ✓ Succeeded Re-Run

Started: a minute ago Entity: Windows\_10\_Template\_hcll cloned  
Duration: 0.246 seconds Action: firefox - 1.0.0  
Exit status: 0 Run as user: nt authority\system

Logs Text Inputs

Search... Run System Info Warning Error Follow logs

```

SYSTEM
1 level="info" msg="Received start action request" actionID="618246d7ff2eff04f58c928a" actionName="firefox"
2 level="info" msg="Sending run update"
3 level="info" msg="Updating run status to running"
4 level="info" msg="Start running action"
RUN
1 & : Le terme C:\Program Files\Mozilla Firefox\firefox.exe n'est pas reconnu comme nom d'applet de commande,
2 fonction, fichier de script ou programme exécutable. Vérifiez l'orthographe du nom, ou si un chemin d'accès existe,
3 vérifiez que le chemin d'accès est correct et réessayez.
4 Au caractère C:\Users\hns\AppData\Local\Temp\run-61828dd3ff2eff04f58c92eb-378983182\firefox.ps1:1 : 3
5 + & 'C:\Program Files\Mozilla Firefox\firefox.exe' $args[0]
6 + ~~~~~
7 + CategoryInfo          : ObjectNotFound: (C:\Program File...fox\firefox.exe:String) [], CommandNotFoundException
8 + FullyQualifiedErrorId : CommandNotFoundException
9
SYSTEM
5 level="info" msg="runStdout read EOF"
6 level="info" msg="runStderr read EOF"
7 level="info" msg="Run succeeded without errors"
8 level="info" msg="Sending run update"
9 level="info" msg="Updating run status to success"
10 level="info" msg="No output file to retrieve"
11 level="info" msg="Finished Run"

```

Fig. 33: Action firefox failed

### 3.4.1 Windows Powershell

In *Windows Powershell*, you can use the **try catch** block to exit with an error if the action fails:

```

try{
  & "C:\Program Files\Mozilla Firefox\firefox.exe"
}catch{
  Write-Error "firefox.exe cannot be started"
  exit 1
}
# If the process fails, the exit code of the action is 1
# Else it is the exit code of the binary firefox.exe

```

You can also use the **automatic variables** to test the action's ending state.

First there is **\$?**, it's used to return a boolean True or False according to the last executed command:

```

Start-Process "C:\Program Files\Mozilla Firefox\firefox.exe"
if(-not $?){
  Write-Error "Action has failed"
  exit 1
}
# If $? == false, the exit code of the action is 1
# Else it is the exit code of the binary firefox.exe

```

Second is **\$LastExitCode**, it's used to return the error code of the **last executed binary**:

```
Start-Process "C:\Program Files\Mozilla Firefox\firefox.exe"  
if($LastExitCode -ne 0){  
    Write-Error "Action has failed"  
    exit $LastExitCode  
}  
# If the $LastExitCode == 4, the exit code of the action is 4  
# Else it is 0
```

### 3.4.2 Windows batch

The variable **%ERRORLEVEL%** can be tested with 0 (success) and 1 (fail):

```
@echo off  
START "C:\Program Files\Mozilla Firefox\firefox.exe"  
IF %ERRORLEVEL% neq 0 (  
    echo "Fail"  
    exit 1  
)  
echo "Success"  
exit 0  
REM If firefox.exe can't start, it will print "Fail" and return the exit code 1  
REM Else "Success" is printed and 0 is returned
```

### 3.4.3 Linux

In Linux you can use the variable **\$?** to test the exit code:

```
#!/bin/bash  
firefox $1  
if [[ $? -ne 0 ]]; then  
    echo "Fail"  
    exit 1  
fi  
echo "Success"  
exit 0  
# If firefox error code == 1, the script prints Fail and returns the exit code 1  
# Else the script prints "Success" and returns the exit code 0
```