

# MISP Workflows

Automation in MISP made easy

Sami Mokaddem

MISP Project

<https://www.misp-project.org/>

Hack.lu 2023



# CONTENT OF THE PRESENTATION

1. Automation in MISP
2. MISP API / PyMISP
3. PubSub channels (ZeroMQ)
4. MISP Workflows
  - ▶ Fundamentals
  - ▶ Demo with examples
  - ▶ Using the system
  - ▶ How it can be extended

# AUTOMATION IN MISP: WHAT ALREADY EXISTS?



## MISP API / PyMISP

- Needs CRON Jobs in place
- Potentially heavy for the server
- Not realtime



## PubSub channels

- After the actions happen: No feedback to MISP
- Tougher to put in place & to share
- Full integration amounts to develop a new tool

# PUBSUB CHANNELS (ZEROMQ) - FUNDAMENTALS

**Objective:** Learn how to setup realtime automation using the ZeroMQ channel

# ZEROMQ CHANNEL - DEMO

- What is ZeroMQ?
  - ▶ *N-to-N Asynchronous message-processing tasks*
  - ▶ *Publisher (MISP) and consumer (scripts)*
- Configuring ZeroMQ in MISP
- Integrating with the ZeroMQ of MISP

# MISP WORKFLOWS - FUNDAMENTALS

**Objective:** Learn how to use the MISP Workflow feature

# AUTOMATION IN MISP: WHAT ALREADY EXISTS?



## MISP API / PyMISP

- Needs CRON Jobs in place
- Potentially heavy for the server
- Not realtime



## PubSub channels

- After the actions happen: No feedback to MISP
  - Tougher to put in place & to share
  - Full integration amounts to develop a new tool
- No way to **prevent** behavior
- Difficult to setup **hooks** to execute callbacks

# WHAT TYPE OF USE-CASES ARE WE TRYING TO SUPPORT?



- **Prevent** default MISP behaviors to happen
  - ▶ Prevent **publication of events** not passing sanity checks
  - ▶ Prevent **querying** third-party **services** with sensitive information
  - ▶ ...
  
- **Hook** specific actions to run callbacks
  - ▶ **Automatically run** enrichment services
  - ▶ Modify data on-the-fly: False positives, enable CTI-Pipeline
  - ▶ Send notifications in a chat rooms
  - ▶ ...

# SIMPLE AUTOMATION IN MISP MADE EASY



- Why?
  - ▶ Everyone loves **simple automation**
  - ▶ **Visual** dataflow programming
  - ▶ Users want **more control**
- How?
  - ▶ **Drag & Drop** editor
  - ▶ Prevent actions **before they happen**
  - ▶ Flexible **Plug & Play** system
  - ▶ Share workflows, **debug** and **replay**

# EXAMPLE OF USE-CASES

- **Notification** on specific actions
  - ▶ New events matching criteria
  - ▶ New users
  - ▶ Automated alerts for high-priority IOCs
- **Extend** existing MISP behavior
  - ▶ Push data to another system
  - ▶ Automatic enrichment
  - ▶ Sanity check to block publishing / sharing
- **Hook** capabilities
  - ▶ Assign tasks and notify incident response team members
  - ▶ Run curation pipeline
- ...

## DISCLAIMER

The Workflow feature is very **powerful**. But also very **dangerous**.

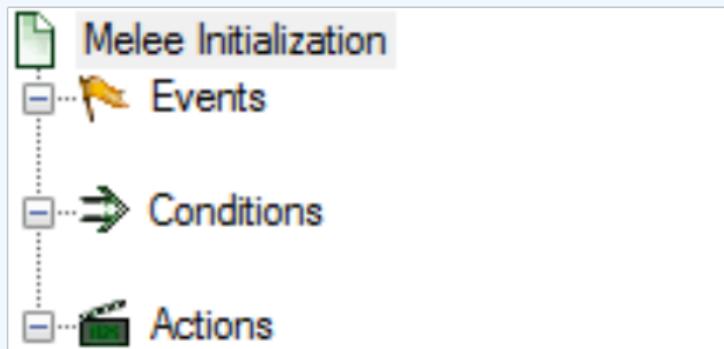


# WORKFLOW - FUNDAMENTALS

**Objective:** Start with the foundation to understand the basics



# HOW DOES IT WORK



1. An **event** happens in MISP
2. Check if all **conditions** are satisfied
3. Execute all **actions**
  - ▶ May prevent MISP to complete its original event

# WHAT KIND OF EVENTS?

## Events

- New MISP Event
- Attribute has been saved
- New discussion post
- New user created
- Query against third-party services
- ...

- ② Supported events in MISP are called **Triggers**
- ② A **Trigger** is associated with **1-and-only-1 Workflow**

# TRIGGERS CURRENTLY AVAILABLE

Currently 10 triggers can be hooked. 3 being Blocking.

## Triggers

List the available triggers that can be listened to by workflows.

Missing a trigger? Feel free to open a Github issue!

Documentation and concepts

« previous

next »

All	attribute	event	object	others	post	user	Blocking	Enabled	Disabled	Trigger name	Scope	Trigger overhead	Run counter	Blocking Workflow	MISP Core format	Workflow ID	Last Update	Debug enabled	Enabled	Actions		
											attribute		83			160	2022-08-03 09:00:41	<input type="checkbox"/>				
											others		1154			162	2022-10-17 12:35:57	<input type="checkbox"/>				
											event		49			175	2022-10-14 13:32:01	<input type="checkbox"/>				
											event		5			182	2022-10-17 09:12:14	<input checked="" type="checkbox"/>				
											event		6			183	2022-10-17 09:01:36	<input checked="" type="checkbox"/>				
											event		126			180	2022-10-13 10:42:53	<input checked="" type="checkbox"/>				
											object		35			161	2022-08-05 07:12:52	<input type="checkbox"/>				
											post		36			176	2022-07-28 13:59:51	<input type="checkbox"/>				
											user		0			181	2022-08-05 07:19:46	<input type="checkbox"/>				
											user		42			158	2022-07-28 14:00:32	<input type="checkbox"/>				

Page 1 of 1, showing 1 records out of 10 total, starting on record 1, ending on 10

# WHAT KIND OF CONDITIONS?

## → Conditions

- A MISP Event is tagged with tlp:red
- The distribution of an Attribute is a sharing group
- The creator organisation is circl.lu
- Or any other **generic** conditions

❓ These are also called **Logic modules**



# WORKFLOW - LOGIC MODULES

- ➔ logic modules: Allow to redirect the execution flow.
  - ▶ IF conditions
  - ▶ Delay execution

All	Action	Logic	misp-module	Custom	Blocking	Enabled	Disabled	Enter value to search	Filter	X	
					Type	Blocking	MISP Core format	misp-module	Custom	Enabled	Actions
<input type="checkbox"/>	Module name				logic	✗	✗	✗	✓	✗	► ⓘ
<input type="checkbox"/>	(Blueprint logic module)				logic	✗	✗	✗	✗	✓	■ ⓘ
<input type="checkbox"/>	.concurrent Task				logic	✗	✗	✗	✗	✓	■ ⓘ
<input type="checkbox"/>	IF :: Distribution				logic	✗	✓	✗	✗	✓	■ ⓘ
<input type="checkbox"/>	Filter :: Generic				logic	✗	✗	✗	✗	✗	► ⓘ
<input type="checkbox"/>	Filter :: Remove filter				logic	✗	✗	✗	✗	✗	► ⓘ
<input type="checkbox"/>	IF :: Generic				logic	✗	✗	✗	✗	✓	■ ⓘ
<input type="checkbox"/>	IF :: Organisation				logic	✗	✓	✗	✗	✓	■ ⓘ
<input type="checkbox"/>	IF :: Published				logic	✗	✓	✗	✗	✓	■ ⓘ
<input type="checkbox"/>	IF :: Tag				logic	✗	✓	✗	✗	✓	■ ⓘ
<input type="checkbox"/>	IF :: Threat Level				logic	✗	✗	✗	✗	✗	► ⓘ

# WHAT KIND OF ACTIONS?



## Actions

- Send an email notification
- Perform enrichments
- Send a chat message on MS Teams
- Attach a local tag
- ...

❓ These are also called **Action modules**

The screenshot shows the configuration interface for the 'Send Mail' action module. At the top, there's a title bar with the module name and a '...' button. Below it is a descriptive text: 'Allow to send a Mail to a list or recipients'. The configuration fields are organized into sections:

- Recipients:** A dropdown menu set to 'All accounts'.
- Mail template subject:** A text input field containing 'I'm the mail subject!'. To its right is a small circular icon with a dot.
- Mail template body:** A text input field containing 'And I'm the body!'.

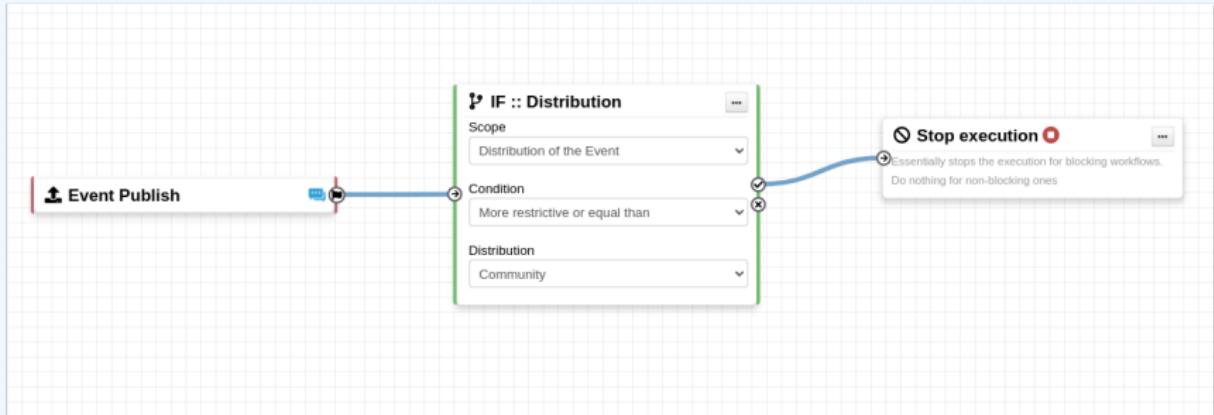
# WORKFLOW - ACTION MODULES

- **action** modules: Allow to execute operations
  - ▶ Tag operations
  - ▶ Send notifications
  - ▶ Webhooks & Custom scripts

All	Action	Logic	misp-module	Custom	Blocking	Enabled	Disabled	Enter value to search	Filter	X
Module name		Type	Blocking	MISP Core format	misp-module	Custom	Enabled	Actions		
<input type="checkbox"/>	* Attach enrichment	action	x	✓	x	x	✓			
<input type="checkbox"/>	Attribute edition operation	action	x	✓	x	x	✓			
<input type="checkbox"/>	Attribute IDS Flag operation	action	x	✓	x	x	✓			
<input type="checkbox"/>	Blueprint action module	action	x	x	x	✓	✓			
<input type="checkbox"/>	* Enrich Event	action	x	✓	x	x	✓			
<input type="checkbox"/>	mattermost	action	x	x	✓	x	✓			
<input type="checkbox"/>	MS Teams Webhook	action	x	x	x	x	✓			
<input type="checkbox"/>	Push to ZMQ	action	x	x	x	x	✓			
<input type="checkbox"/>	Send Log Mail	action	x	x	x	x	x			
<input type="checkbox"/>	Send Mail	action	x	x	x	x	✓			
<input type="checkbox"/>	> Splunk HEC export	action	x	✓	x	x	x			
<input type="checkbox"/>	Stop execution	action	✓	x	x	x	✓			
<input type="checkbox"/>	Tag operation	action	x	✓	x	x	✓			
<input type="checkbox"/>	testaction	action	x	x	✓	x	✓			
<input type="checkbox"/>	Webhook	action	x	x	x	x	✓			

# WHAT IS A MISP WORKFLOW?

- Sequence of all nodes to be executed in a specific order
- Workflows can be enabled / disabled
- A Workflow is associated to **1-and-only-1 trigger**



# SOURCES OF WORKFLOW MODULES (o)

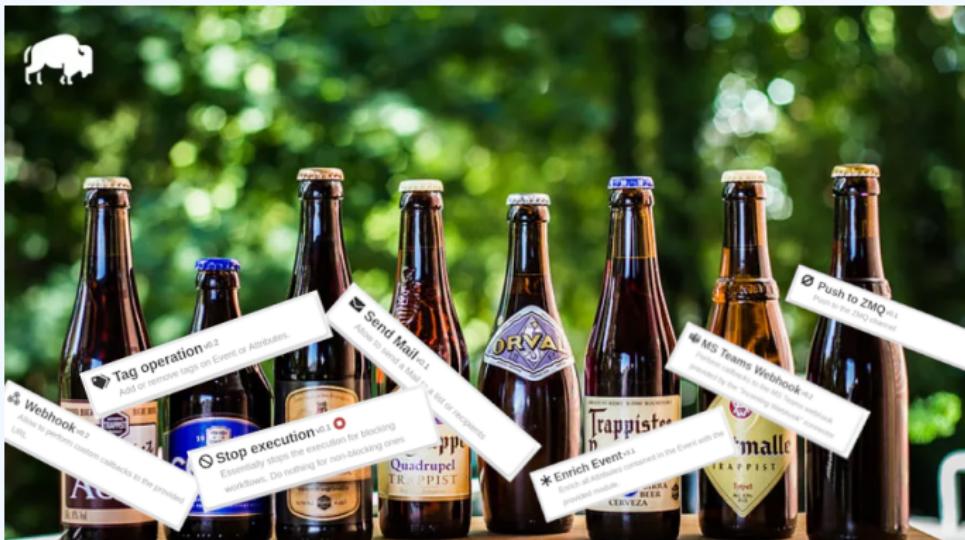
Currently 36 built-in modules.

- **Trigger** module (11): built-in **only**
  - ▶ Get in touch if you want more
- **Logic** module (10): built-in & **custom**
- **Action** module (15): built-in & **custom**

# SOURCES OF WORKFLOW MODULES (1)

## ■ Built-in **default** modules

- ▶ Part of the MISP codebase
- ▶ Get in touch if you want us to increase the selection (or merge PR!)



# SOURCES OF WORKFLOW MODULES (2)

## User-defined **custom** modules

- Written in PHP
- Extend existing modules
- MISP code reuse

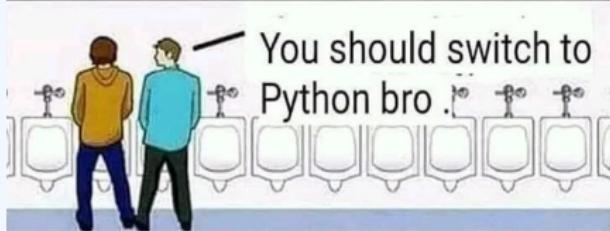


# SOURCES OF WORKFLOW MODULES (3)

Modules from the

misp-module<sup>⊗⊗</sup>

enrichment service



- Written in Python
- Can use any python libraries
- Plug & Play

## DEMO BY EXAMPLES

- WF-1. Send an email to **all** when a new event has been pulled
  
- WF-2. Block queries on 3rd party services when **tlp:red** or **PAP:red**
  - ▶ **tlp:red**: For the eyes and ears of individual recipients only
  - ▶ **PAP:RED**: Only passive actions that are not detectable from the outside

# WORKFLOW - GETTING STARTED

**Objective:** How to install & configure workflows



## 2.4.160 Epic summer release



iglocska released this 08 Aug 2022



v2.4.160



71d4e2c



1. Update your MISP server
2. Update all your sub-modules



# GETTING STARTED WITH WORKFLOWS (2)

Review MISP settings:

1. Make sure MISP.background\_jobs is turned on
2. Make sure workers are up-and-running and healthy
3. Turn the setting Plugin.Workflow\_enable on

The screenshot shows the 'Plugin settings' tab selected in the top navigation bar. Below it, a table lists various configuration items. The 'Workflow' section is highlighted with a blue border. It contains a single row with the following data:

Recommended	Plugin.Workflow_enable	true	Enable/disable workflow feature

# GETTING STARTED WITH WORKFLOWS (3)

Review MISP settings:

4. [optional:misp-module] Turn the setting `Plugin.Action_services_enable` on

Overview	MISP settings (20 ▲)	Encryption settings (7 ▲)	Proxy settings (5)	Security settings (8 ▲)	Plugin settings (465 ▲)	SimpleBackgroundJobs settings (11 ▲)	Diagnos
Enrichment							Filter the table(s) below
Import							
Export							
Action							
Critical	Plugin.Action_services_enable	true		Enable/disable the action services			
Recommended	Plugin.Action_services_url	http://host.docker.internal		The url used to access the action services. By default, it is accessible at http://127.0.0.1:6666			
Recommended	Plugin.Action_services_port	6677		The port used to access the action services. By default, it is accessible at 127.0.0.1:6666			
Recommended	Plugin.Action_timeout	10		Set a timeout for the action services			Value not set.

# GETTING STARTED WITH WORKFLOWS (4)

If you wish to use action modules from misp-module, make sure to have:

- The latest update of misp-module
  - ▶ There should be an `action_mod` module type in `misp-modules/misp_modules/modules`
- Restarted your misp-module application

---

```
1 # This command should show all 'action' modules
2 $ curl -s http://127.0.0.1:6666/modules | \
3 jq '.[] | select(.meta.module-type == "action") | \
4 {name: .name, version: .meta.version}'
```

---

# GETTING STARTED WITH WORKFLOWS (5)

Everything is ready?

Let's see how to build a workflow!



# CREATING A WORKFLOW WITH THE EDITOR

1. Prevent event publication if **tlp:red** tag
2. Send a mail to admin@admin.test about potential data leak
3. Otherwise, send a notification on **Mattermost, MS Teams, Telegram, ...**

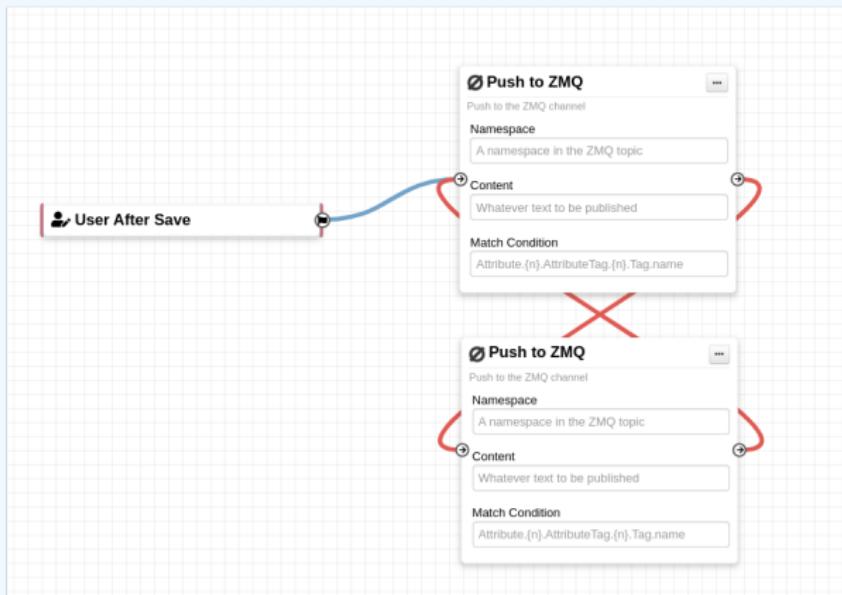
# CONSIDERATIONS WHEN WORKING WITH WORKFLOWS

**Objective:** Overview of some common pitfalls

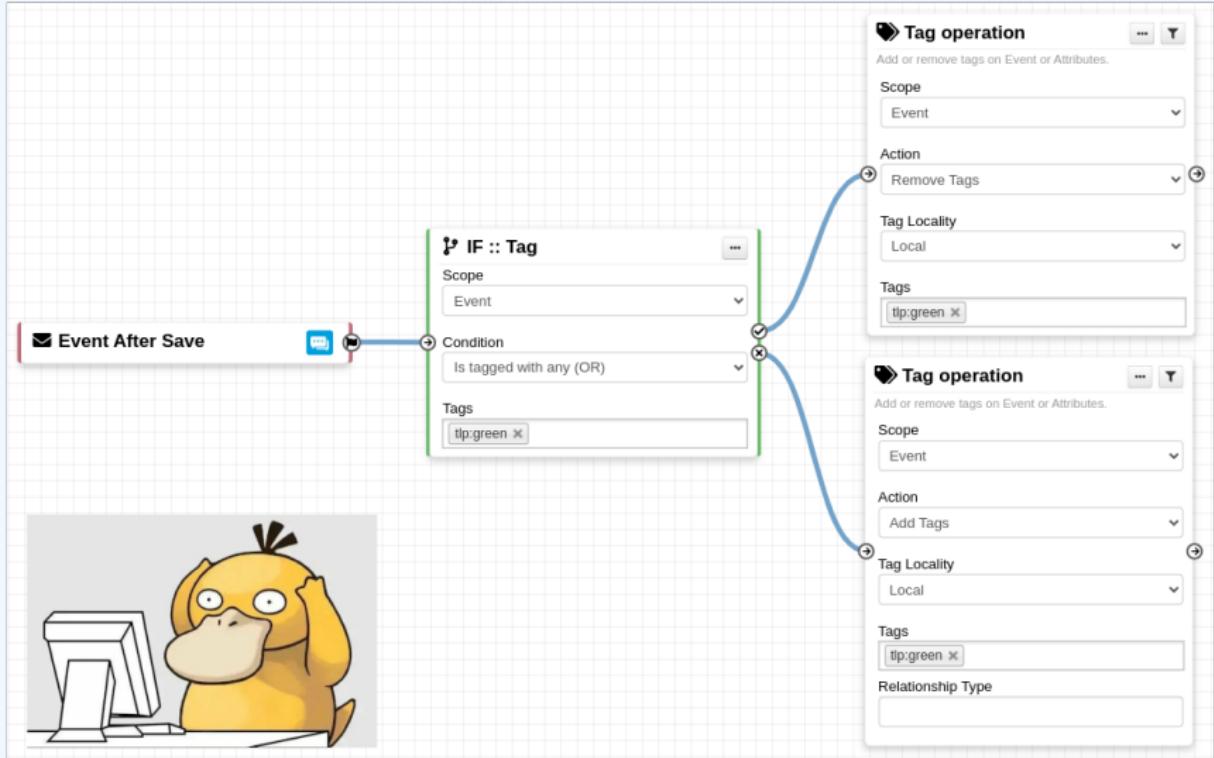


# WORKING WITH THE EDITOR - OPERATIONS NOT ALLOWED

Execution loop are not authorized



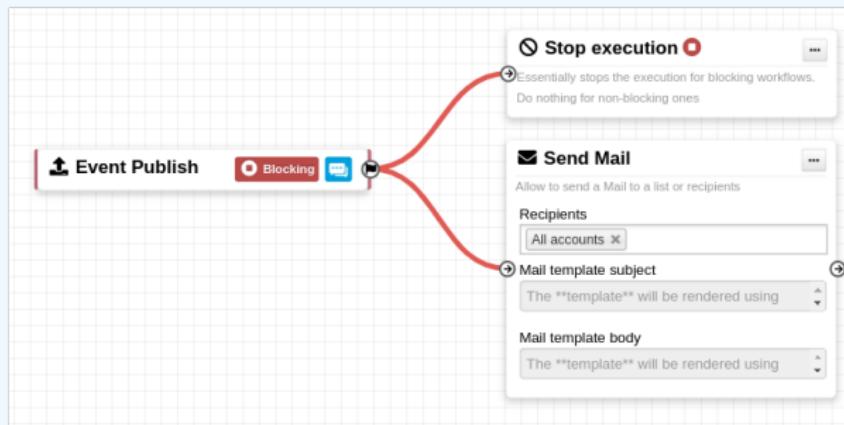
# RECURSIVE WORKFLOWS



⚠ Recursion: If an action re-runs the workflow

# WORKING WITH THE EDITOR - OPERATIONS NOT ALLOWED

Multiple connections from the same output

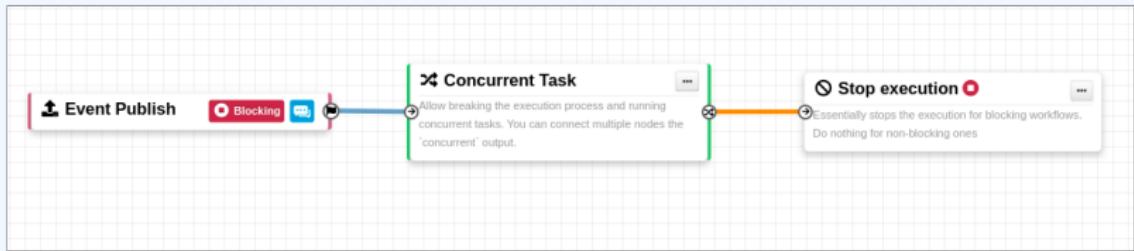


- Execution order not guaranteed
- Confusing for users

# WORKING WITH THE EDITOR

Cases showing a warning:

- **Blocking modules**  in a  **Non blocking** workflow
- **Blocking modules**  after a **concurrent tasks** module



# ADVANCED USAGE

**Objective:** Overview of Blueprints, Data format and Filtering

# WORKFLOW BLUEPRINTS



1. Blueprints allow to **re-use parts** of a workflow in another one
2. Blueprints can be saved, exported and **shared**

**Debugging webhook** v1656059209

9ff210dd-ee7e-49c8-a5af-10cd42cdadb6

Default: **X**

Blueprint Content: **1 node**

 1

Webhook module pre-configured for debugging purposes

Blueprints sources: MISP/misp-workflow-blueprints repository

- Block sharing if any attributes have the PAP:RED or tlp:red tag
- Curation pipeline
- Enrich data from 3rd-party

# BLOCKING AND NON-BLOCKING

Two types of workflows:

## ✗ Blocking Workflows

- ▶ Can prevent / block the original event to happen
- ▶ If a **blocking module** ✗ blocks the action
- ▶ Example: Event publish, Enrichment Before Query, ...

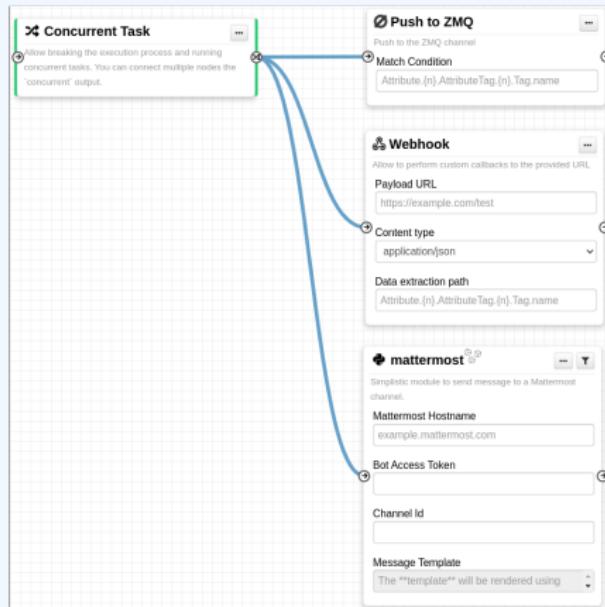
## ✓ Non blocking Workflows execution outcome has no impact

- ▶ No way to prevent something that happened in the past
- ▶ Example: Event after-save, Attribute after-save, ...



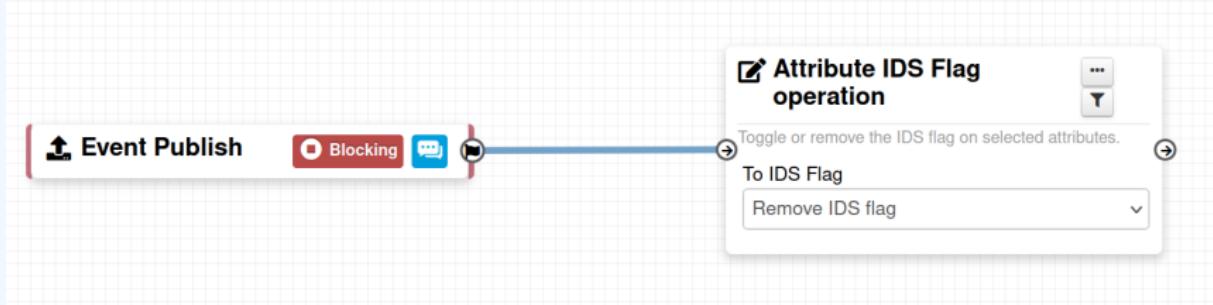
# LOGIC MODULE: CONCURRENT TASK

- Logic module allowing **multiple output** connections
- **Postpone the execution** for remaining modules
- Convert Blocking → Non blocking



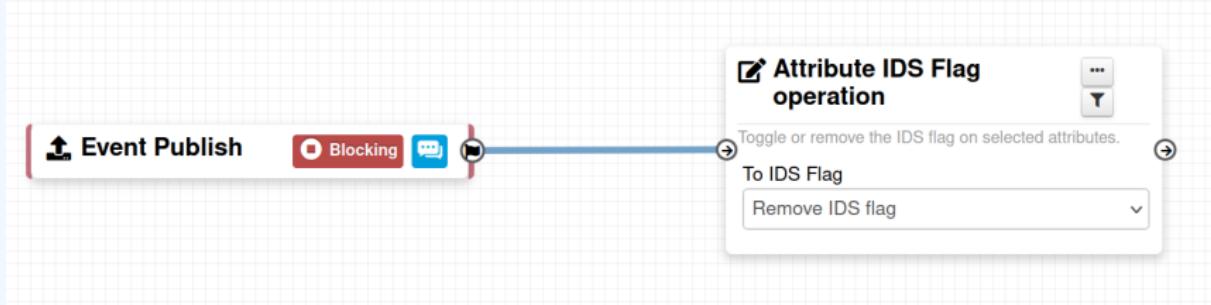
# FILTERING DATA ON WHICH TO APPLY A MODULE

What happens when an Event is about to be published?



# FILTERING DATA ON WHICH TO APPLY A MODULE

What happens when an Event is about to be published?

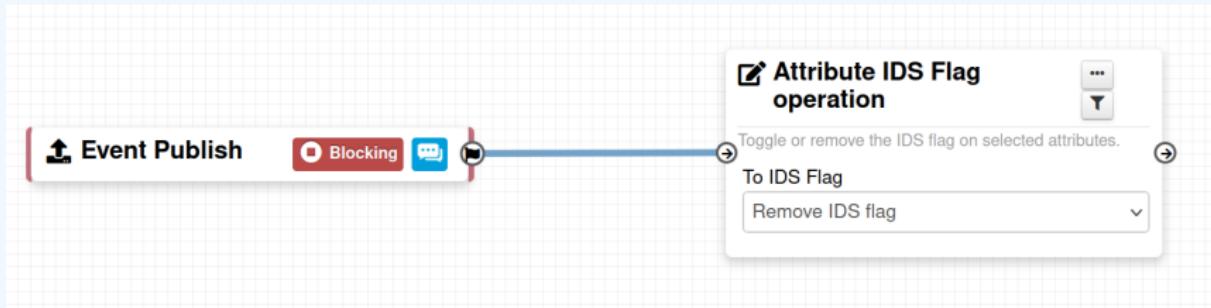


All Attributes get their to\_ids turned off.

- ❓ How could we force that action, only on Attribute of type comment?

# FILTERING DATA ON WHICH TO APPLY A MODULE

What happens when an Event is about to be published?



All Attributes get their `to_ids` turned off.

❓ How could we force that action, only on Attribute of type comment?

→ Hash path filtering!

# HASH PATH FILTERING

Hash path filtering can be used to **filter** data **on a node** or on the **execution path**.

**Node Filtering**

Element selector  
Event.\_AttributeFlattened.{n}

Value  
domain

Operator  
Equals

Hash Path  
type

**Save** **Close**

**T Filter :: Generic**

Generic data filtering block. The module filters incoming data and forward the matching data to its output.

Filtering Label  
Label A

Data selector  
Event.\_AttributeFlattened.{n}

Value  
tip:red

Operator  
In

Hash path  
Tag.{n}.name

# DATA FORMAT IN WORKFLOWS



Blocking



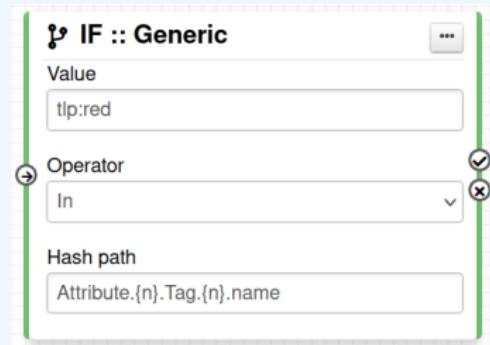
- In most cases, the format is the **MISP Core format**
  - ▶ Attributes are **always encapsulated** in the Event or Object
- But has **additional properties**
  - ▶ Additional key **\_AttributeFlattened**
  - ▶ Additional key **\_allTags**
  - ▶ Additional key **inherited** for Tags

# HASH PATH FILTERING (1)

Filtering and checking conditions using hash path expression.

```
1 $path_expression = '{n}[name=fred].id';
2 $users = [
3     {'id': 123, 'name': 'fred', 'surname': 'bloggs'},
4     {'id': 245, 'name': 'fred', 'surname': 'smith'},
5     {'id': 356, 'name': 'joe', 'surname': 'smith'},
6 ];
7 $ids = Hash::extract($users, $path_expression);
8 // => $ids will be [123, 245]
```

```
{
    "Attribute": [
        {
            "type": "domain",
            "value": "cti-summit.org",
            "Tag": [
                {
                    "name": "tlp:red",
                    "colour": "#CC0033"
                }
            ]
        }
    ]
}
```



# HASH PATH FILTERING - EXAMPLE

```
1  {
2      "Event": {
3          "uuid": ...
4          "timestamp": ...
5          "distribution": 1,
6          "published": false,
7          "Attribute": [
8              {
9                  "type": "ip-src",
10                 "value": "8.8.8.8", ...
11             },
12             {
13                 "type": "domain",
14                 "value": "misp-project.org", ...
15             }
16         ],
17         ...
18     }
19 }
```

1. Access Event distribution
  - ▶ Event.distribution

# HASH PATH FILTERING - EXERCISE (1)

---

```
1  {
2      "Event": {
3          "uuid": ...,
4          "distribution": 1,
5          "published": false,
6          "Attribute": [
7              {
8                  "type": "ip-src",
9                  "value": "8.8.8.8", ...
10             },
11             {
12                 "type": "domain",
13                 "value": "misp-project.org", ...
14             }
15         ],
16         ...
17     }
18 }
```

---

## 2. Access Event published state

# HASH PATH FILTERING - EXERCISE (1)

---

```
1  {
2      "Event": {
3          "uuid": ...,
4          "distribution": 1,
5          "published": false,
6          "Attribute": [
7              {
8                  "type": "ip-src",
9                  "value": "8.8.8.8", ...
10             },
11             {
12                 "type": "domain",
13                 "value": "misp-project.org", ...
14             }
15         ],
16         ...
17     }
18 }
```

- 
2. Access Event published state
    - ▶ Event.published

## HASH PATH FILTERING - EXERCISE (2)

```
1  {
2      "Event": {
3          "uuid": ...
4          "distribution": 1,
5          "published": false ,
6          "Attribute": [
7              {
8                  "type": "ip-src",
9                  "value": "8.8.8.8", ...
10             },
11             {
12                 "type": "domain",
13                 "value": "misp-project.org", ...
14             }
15         ],
16         ...
17     }
18 }
```

- 
3. Access all Attribute types
    - ▶ Hint: Use **{n}** to loop

## HASH PATH FILTERING - EXERCISE (2)

```
1  {
2      "Event": {
3          "uuid": ...
4          "distribution": 1,
5          "published": false ,
6          "Attribute": [
7              {
8                  "type": "ip-src",
9                  "value": "8.8.8.8", ...
10             },
11             {
12                 "type": "domain",
13                 "value": "misp-project.org", ...
14             }
15         ],
16         ...
17     }
18 }
```

### 3. Access all Attribute types

- ▶ Hint: Use **{n}** to loop
- ▶ Event.Attribute.{n}.type

## HASH PATH FILTERING - EXERCISE (3)

---

```
1  {
2      "Event": {
3          "Attribute": [
4              {
5                  "type": "ip-src",
6                  "value": "8.8.8.8",
7                  "Tag": [
8                      {
9                          "name": "PAP:AMBER", ...
10                     }
11                 ],
12             }
13         ],
14     ...
15 }
16 }
```

---

3. Access all Tags attached to Attributes

## HASH PATH FILTERING - EXERCISE (3)

```
1  {
2      "Event": {
3          "Attribute": [
4              {
5                  "type": "ip-src",
6                  "value": "8.8.8.8",
7                  "Tag": [
8                      {
9                          "name": "PAP:AMBER", ...
10                     }
11                 ],
12             }
13         ],
14     ...
15 }
16 }
```

### 3. Access all Tags attached to Attributes

- ▶ Event.Attribute.{n}.Tag.{n}.name

## HASH PATH FILTERING - EXERCISE (4)

```
1  {
2      "Event": {
3          "Tag": [
4              {
5                  "name": "tlp:green", ...
6              }
7          ], ...
8          "Attribute": [
9              {
10                 "value": "8.8.8.8",
11                 "Tag": [
12                     {
13                         "name": "PAP:AMBER", ...
14                     }
15                 ], ...
16             }
17         ],
18     }
19 }
```

4. Access all Tags attached to Attributes and from the Event
  - ▶ Hint: Use `_allTags` to access **all** tags

## HASH PATH FILTERING - EXERCISE (4)

```
1  {
2      "Event": {
3          "Tag": [
4              {
5                  "name": "tlp:green", ...
6              }
7          ], ...
8          "Attribute": [
9              {
10                 "value": "8.8.8.8",
11                 "Tag": [
12                     {
13                         "name": "PAP:AMBER", ...
14                     }
15                 ], ...
16             }
17         ],
18     }
19 }
```

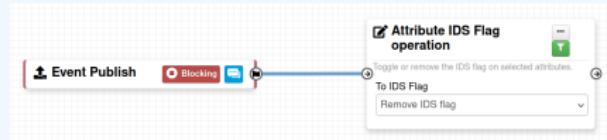
4. Access all Tags attached to Attributes and from the Event
  - ▶ `Event.Attribute.{n}._allTags.{n}.name`

## HASH PATH FILTERING - EXERCISE (4)

```
1  {
2      "Event": {
3          "Tag": [...],
4          "Attribute": [
5              {
6                  "value": "8.8.8.8",
7                  "_allTags": [
8                      {
9                          "name": "tlp:green",
10                         "inherited": true,
11                     },
12                     {
13                         "name": "PAP:AMBER",
14                         "inherited": false,
15                     }
16                 ],
17             }
18             ...
19 }
```

4. Access all Tags attached to Attributes and from the Event
  - ▶ `Event.Attribute.{n}._allTags.{n}.name`

# FITLERING DATA ON WHICH TO APPLY A MODULE



## Node Filtering

Element selector  
Event.\_AttributeFlattened.{n}

Value  
comment

Operator  
In

Hash Path  
type

# FITLERING DATA ON WHICH TO APPLY A MODULE



## Node Filtering

Element selector

Event.\_AttributeFlattened.{n}

Select elements on which  
to apply the filtering

Value

comment

Fixed value

Operator

In

Comparison operator

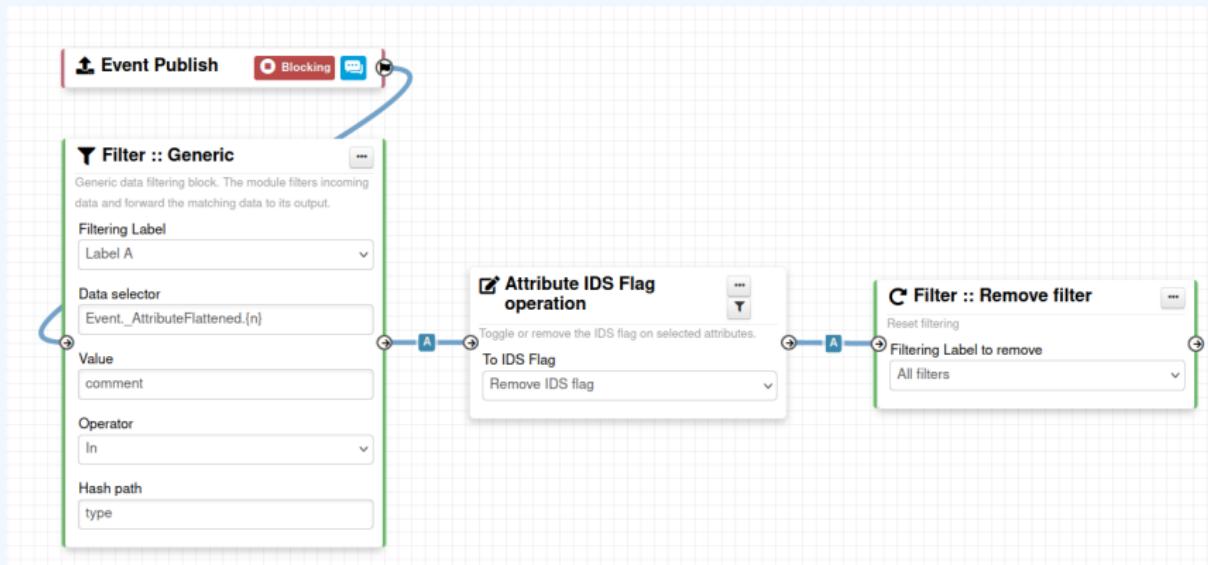
Hash Path

type

Data point to get the value

# FITLERING DATA ON WHICH TO APPLY ON MULTIPLE MODULES

New feature as of **v2.4.171** allows setting filters on a path.



# EXERCICES

# EXERCISES

Try to build it in the training instance. **Do not save it!**

1. PAP:RED and tlp:red blocking
2. Replace tlp:white by tlp:clear
3. Remove to\_ids flag for attribute having a match in hashlookup
4. Attach tag on attribute having a low value (<50) in bgp ranking

# **DEBUGGING**

# DEBUGGING WORKFLOWS: LOG ENTRIES

- Workflow execution is logged in the application logs:
  - ▶ `/admin/logs/index`
  - ▶ Note: Might be phased out as its too verbose
- Or stored on disk in the following file:
  - ▶ `/app/tmp/logs/workflow-execution.log`

## Logs

The screenshot shows a log viewer with a header bar containing links: 'Emails', 'Authentication issues', 'MISP Update results', 'Setting changes', and 'Warnings and errors'. Below this is a table with the following columns: 'Id ↑', 'Email', 'Org', 'Created', 'Model', 'Model ID', 'Action', and 'Title'. The first row contains data for log entry 49146, and the second row contains data for log entry 49144. The rows alternate in color.

Id ↑	Email	Org	Created	Model	Model ID	Action	Title
49146	SYSTEM	SYSTEM	2022-08-01 07:34:40	Workflow	162	execute_workflow	Finished executing workflow for trigger `enrichment-before-query` (162). Outcome: success
49144	SYSTEM	SYSTEM	2022-08-01 07:34:39	Workflow	162	execute_workflow	Started executing workflow for trigger `enrichment-before-query` (162)

# DEBUGGING WORKFLOWS: DEBUG MODE

- The  Debug Mode: On can be turned on for each workflows
- Each nodes will send data to the provided URL
  - Configure the setting: Plugin.Workflow\_debug\_url
- Result can be visualized in
  - offline:** tools/misp-workflows/webhook-listener.py
  - online:** requestbin.com or similar websites

LIVE	PAUSE	<input type="text"/> Type to search...
<b>Today</b>		
2:25:10 pm	POST	/end?outcome=blocked
2:25:09 pm	POST	/exec/stop-execution?result=success
2:25:09 pm	POST	/exec/tag-if?result=success
2:25:08 pm	POST	/init?type=blocking

# DEBUGGING MODULES: STATELESS EXECUTION

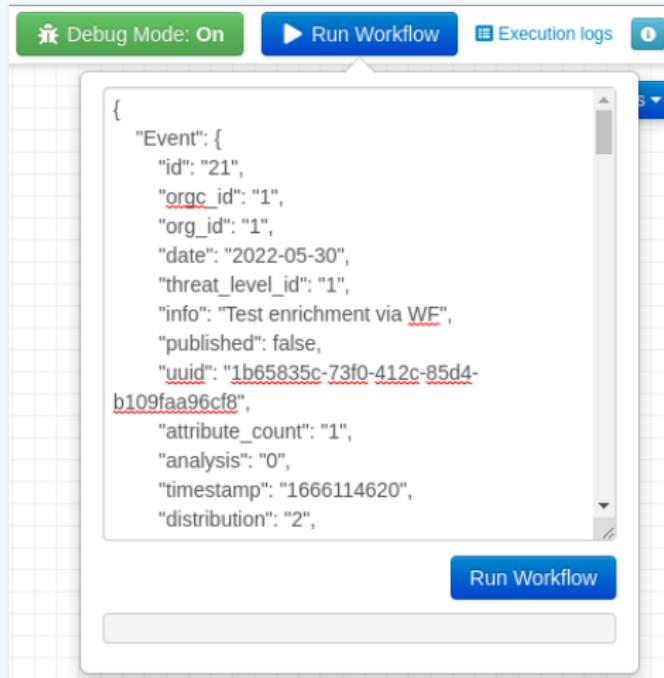
## ■ Test custom modules with custom input

### Stateless module execution

Module parameters	Input data
Payload URL <input type="text" value="https://localhost:8443"/>	<input type="checkbox"/> Convert input data into MISP core format Module Input Data <pre>{   "foo": "bar" }</pre>
Content type <input type="text" value="application/json"/>	
Data extraction path <input type="text" value="Attribute.{n}.AttributeTag.{n}.Tag.name"/>	
<button type="button">Execute module</button>	
Execution result: <span style="background-color: #2e7131; color: white; padding: 2px 5px;">200 [56 ms]</span>	

# DEBUGGING MODULES: RE-RUNNING WORKFLOWS

- Try workflows with custom input
- Re-run workflows to ease debugging

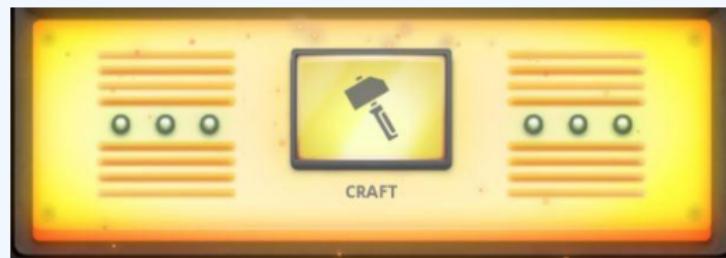


# DEBUGGING OPTIONS

- Workflow **execution and outcome**
- Module **execution and outcome**
- **Live** workflow debugging with module inspection
- **Re-running/testing** workflows with custom data
- **Stateless** module execution



# EXTENDING THE SYSTEM



# CREATING A NEW MODULE IN PHP



- **app/Lib/WorkflowModules/action/[module\_name].php**
- Designed to be easily extended
  - ▶ Helper functions
  - ▶ Module configuration as variables
  - ▶ Implement runtime logic
- Main benefits
  - ▶ Fast
  - ▶ Re-use existing functionalities
  - ▶ No need for misp-modules

# CREATING A NEW MODULE IN PHP

```
app > Lib > WorkflowModules > action > 🏷 Module_blueprint_action_module.php > ...
1  <?php
2  include_once APP . . . 'Model/WorkflowModules/WorkflowBaseModule.php';
3
4  class Module_blueprint_action_module extends WorkflowBaseModule
5  {
6      public $is_blocking = false;
7      public $disabled = true;
8      public $id = 'blueprint-action-module';
9      public $name = 'Blueprint action module';
10     public $description = 'Lorem ipsum dolor, sit amet consectetur adipisicing elit.';
11     public $icon = 'shapes';
12     public $inputs = 1;
13     public $outputs = 1;
14     public $params = [];
15
16     public function exec(array $node, WorkflowRoamingData $roamingData, array &$errors = [])
17     : bool
18     {
19         parent::exec($node, $roamingData, $errors);
20         // If $this->is_blocking == true, returning `false` will stop the execution.
21         $errors[] = __('Execution stopped');
22         return false;
23     }
}
```

# CREATING A NEW MODULE IN PYTHON



- Similar to how other misp-modules are implemented
  - ▶ Helper functions
  - ▶ Module configuration as variables
  - ▶ Implement runtime logic
- Main benefits
  - ▶ Easier than PHP
  - ▶ Lots of libraries for integration

# CREATING A NEW MODULE IN PYTHON

```
home > sami > git > misp-modules > misp_modules > modules > action_mod > testaction.py > ...
1 > import json
2
3
4 misperrors = {'error': 'Error'}
5
6 # config fields that your code expects from the site admin
7 moduleconfig = {
8     'foo': {
9         'type': 'string',
10        'description': 'blablabla',
11        'value': 'xyz'
12    },
13    'bar': {
14        'type': 'string',
15        'value': 'meh'
16    }
17 };
18
19 # blocking modules break the execution of the chain of actions (such as publishing)
20 blocking = False
21
22 # returns either "boolean" or "data"
23 # Boolean is used to simply signal that the execution has finished.
24 # For blocking modules the actual boolean value determines whether we break execution
25 returns = 'boolean'
26
27 moduleinfo = {'version': '0.1', 'author': 'Andras Iklody',
28               'description': 'This module is merely a test, always returning true. Triggers on event publishing.',
29               'module-type': ['action']}
30
31
32 def handler(q=False):
33     if q is False:
34         return False
35     result = json.loads(q) # noqa
36     output = result # Insert your magic here!
37     r = {"data": output}
38     return r
```

# SHOULD I MIGRATE TO MISP WORKFLOWS

I have automation in place using the API / ZMQ. Should I move to Workflows?

- I (have/am planning to create) a curation pipeline using the API, should I port them to workflows?
  - ▶ **No** in general, but WF can be used to start the curation process
- What if I want to **block** some actions
  - ▶ Put the blocking logic in the WF, the remaining outside
- Currently, workflows with **lots of node are not encouraged**
- Bottom line is **Keep it simple**

# FUTURE WORKS

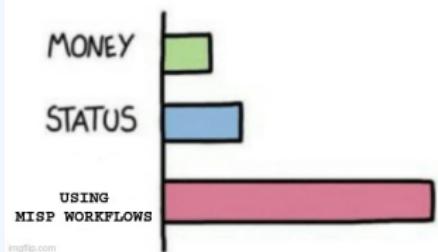
- More 🎥 modules
- More ➔ modules
- More 🚨 triggers
- More documentation
- Recursion prevention system
- On-the-fly data override?



# FINAL WORDS

- Designed to **quickly** and **cheaply** integrate MISP in CTI pipelines
- **Beta** Feature unlikely to change.  
But still..
- Waiting for feedback!
  - ▶ New triggers?
  - ▶ New modules?
  - ▶ What's achievable

WHAT GIVES PEOPLE  
FEELINGS OF POWER



imgflip.com