

AUTOMATION WITH MISP WORK- FLOWS

A NEW WAY TO SUPPORT YOUR CTI PIPELINES

SAMI MOKADDEM

MISP PROJECT

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

CTIS 2022

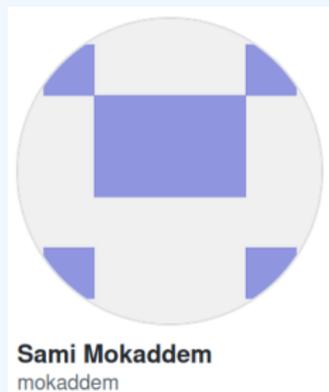


MISP
Threat Sharing

CTIS-2022

\$ whoami

- Working @ circl.lu
- Part of the MISP-Project team
- Adding easter eggs for the past 4 years



Event graph viewer editor #3063

 Merged [adulau](#) merged 27 commits into [MISP:2.4](#) from [mokaddem:ref_graph](#)  on 23 Mar 2018

**Belgian roads
are terrible**

Belgians

**The Belgian
Federal
Government sucks**

Belgians

**The French
invented fries**

**YOU BETTER WATCH
YOUR MOUTH**

WHAT PROBLEMS ARE WE TRYING TO TACKLE?

- **Prevent** default MISP behaviors to happen
- **Hook** specific actions to run callbacks
- Use-cases:
 - ▶ Prevent publication of events not passing sanity checks
 - ▶ Prevent querying thrid-party services with sensitive information
 - ▶ Send notifications in a chat rooms
 - ▶ And much much more...

WHAT ALREADY EXISTS IN MISP?



MISP API / PyMISP

- Needs CRON Jobs in place
- 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



■ Why?

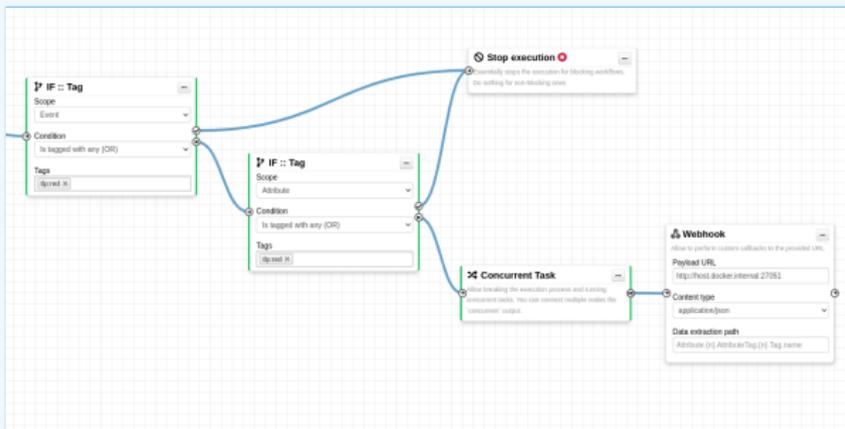
- ▶ Everyone loves **simple automation**
- ▶ **Visual** dataflow programming
- ▶ Users want **more control**

■ How?

- ▶ **Drag & Drop** editor
- ▶ Prevent actions **before they happen**
- ▶ Flexible **Plug & Play** system
- ▶ **Share** workflows, **debug** and **replay**

CONTENT OF THE PRESENTATION

- MISP Workflows fundamentals
- Demo by examples
- Get started
- Using the system & how it can be extended



WORKFLOW - FUNDAMENTALS



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

Events

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

In MISP Workflow terminology, supported events are called **Triggers**

Conditions

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

In MISP Workflow terminology, these are also called **Logic modules**



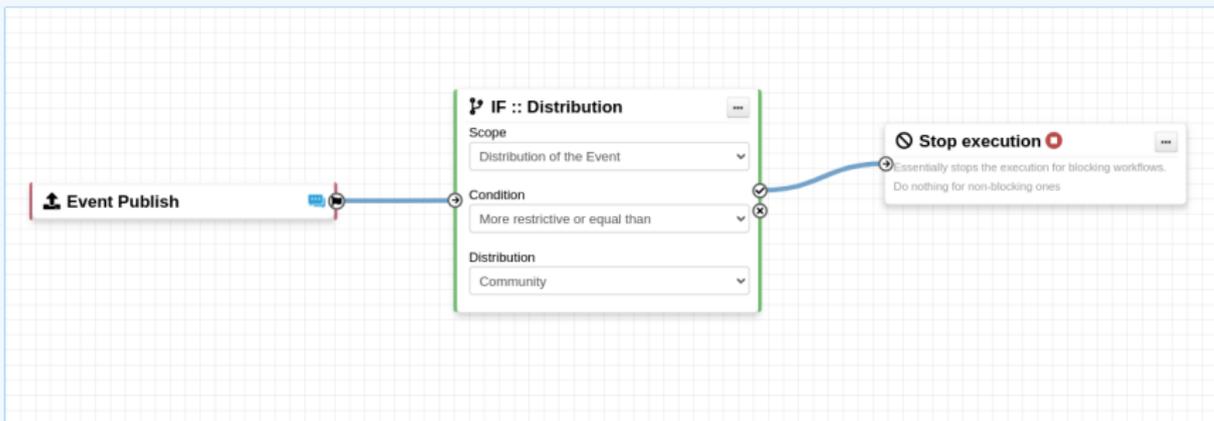
Actions

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

In MISP Workflow terminology, these are also called **Action modules**

WHAT IS A MISP WORKFLOW?

- Sequence of all nodes to be executed in the specified order
- Basically the whole connected graph.
- Workflows can be enabled / disabled
- Workflows are always linked to a **trigger**



WORKFLOW EXECUTION FOR EVENT PUBLISH



An Event is about to be published

- ▶ The workflow for the event-publish trigger starts



Conditions are evaluated



Actions are executed

- ▶ **success:** Continue the publishing action

```
execute_workflow Finished executing workflow for trigger `event-publish` (180). Outcome: success
```

- ▶ **failure | blocked:** Stop publishing and log the reason

```
execute_workflow Execution stopped.
```

```
Node `stop-execution` (8) from Workflow `Workflow for trigger event-publish` (180) returned the following error: Execution stopped
```

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

Regular Workflows execution outcome has no impact

- ▶ **Blocking modules** No way to prevent something that has already happened



WORKFLOW - ACTION MODULES

-  **action** modules: Allow to executes operations or custom scripts
 - ▶ Tag operations
 - ▶ Send notifications
 - ▶ Webhooks

		All	Action	Logic	misp-module	Custom	Blocking	Enabled	Disabled	Enter value to search	Filter	X
<input type="checkbox"/>	Module name	Type	Blocking	MISP Core format	misp-module	Custom	Enabled	Actions				
<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 Mail	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	✓	 				

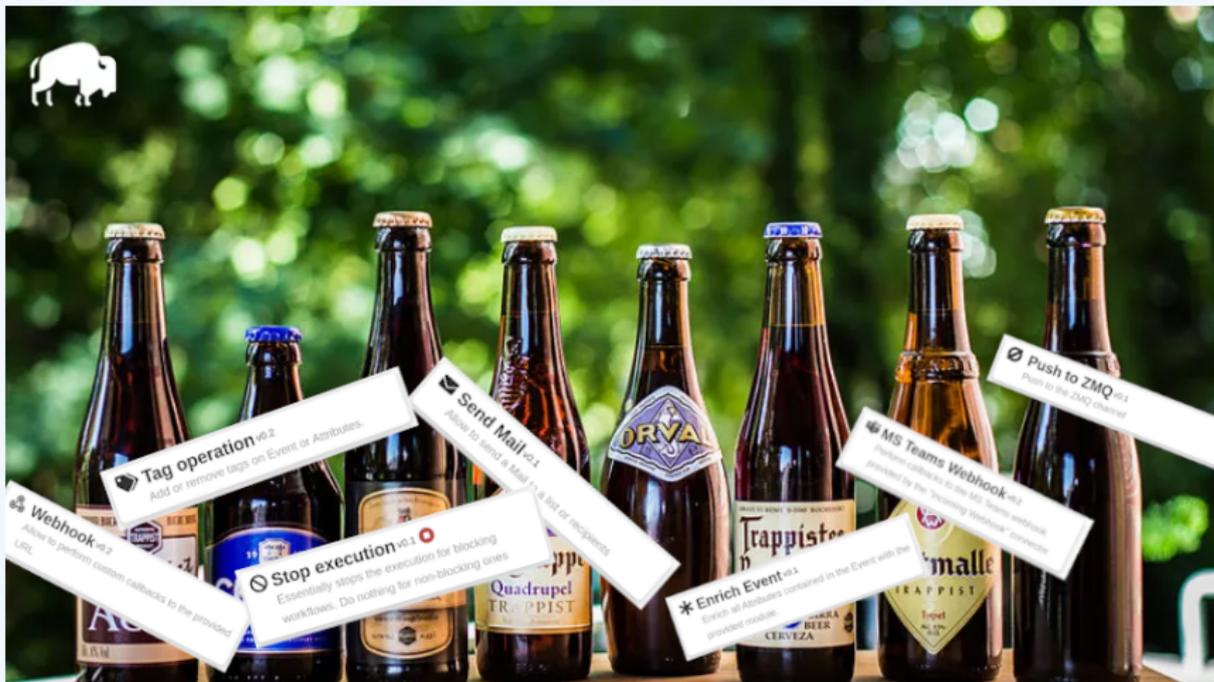
- ➡ **logic** modules: Allow to redirect the execution flow.
 - ▶ IF conditions
 - ▶ Delay execution

All	Action	Logic	misp-module	Custom	Blocking	Enabled	Disabled	Enter value to search	Filter	X	
<input type="checkbox"/>		Module name			Type	Blocking	MISP Core format	misp-module	Custom	Enabled	Actions
<input type="checkbox"/>		Blueprint logic module			logic	×	×	×	✓	×	▶ 🔍
<input type="checkbox"/>		Concurrent Task			logic	×	×	×	×	✓	■ 🔍
<input type="checkbox"/>		IF :: Distribution			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	×	✓	×	×	✓	■ 🔍

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!



SOURCES OF WORKFLOW MODULES (2)

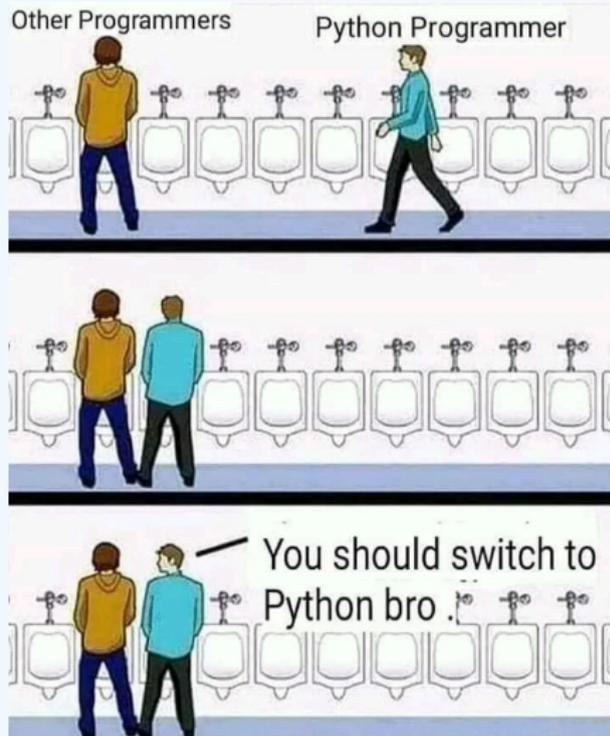
User-defined **custom** modules

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



SOURCES OF WORKFLOW MODULES (3)

Modules from the `misp-module`  enrichment service



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

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!](#)

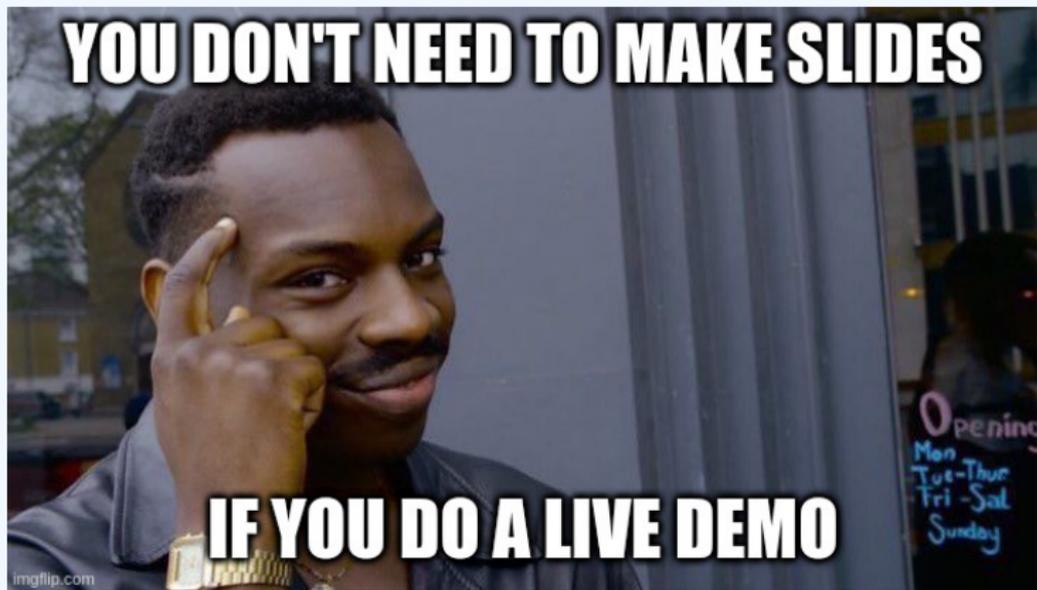
[Documentation and concepts](#)

« 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 After Save	attribute	high 	83	×	✓	160	2022-08-03 09:00:41	<input type="checkbox"/>	×	   
 Enrichment Before Query	others	low 	1154	✓	✓	162	2022-10-17 12:35:57	<input type="checkbox"/>	✓	   
 Event After Save	event	high 	49	×	✓	175	2022-10-14 13:32:01	<input type="checkbox"/>	✓	   
 Event After Save New	event	low 	5	×	✓	182	2022-10-17 09:12:14	<input checked="" type="checkbox"/>	✓	   
 Event After Save New From Pull	event	low 	6	×	✓	183	2022-10-17 09:01:36	<input checked="" type="checkbox"/>	✓	   
 Event Publish	event	low 	126	✓	✓	180	2022-10-13 10:42:53	<input checked="" type="checkbox"/>	✓	   
 Object After Save	object	high 	35	×	✓	161	2022-08-05 07:12:52	<input type="checkbox"/>	×	   
 Post After Save	post	low 	36	×	×	176	2022-07-28 13:59:51	<input type="checkbox"/>	×	   
 User After Save	user	low 	0	×	×	181	2022-08-05 07:19:46	<input type="checkbox"/>	×	   
 User Before Save	user	low 	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



WORKFLOW - GETTING STARTED

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

Overview	MISP settings (20 ▲)	Encryption settings (7 ▲)	Proxy settings (5)	Security settings (8 ▲)	Plugin settings (465 ▲)	SimpleBackgroundJobs settings (11 ▲)	Diagnos
Enrichment	<input type="text" value="Filter the table(s) below"/>						
Import							
Export							
Action							
Cortex							
Sightings							
<u>Workflow</u>							
Recommended	Plugin.Workflow_enable	true	Enable/disable workflow feature				

GETTING STARTED WITH WORKFLOWS (3)

[optional] Wanna enjoy `misp-module` ?

1. 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 ▲) Diagnostics

Enrichment

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.

1. Go to the list of modules
 - ▶ Administration > Workflows > List Modules
 - ▶ or /workflows/moduleIndex
2. Make sure **default** modules are loaded
3. [optional:misp-module] Make sure **misp-module** modules are loaded

Everything is ready?

Let's see how to build a workflow!

CREATING A WORKFLOW WITH THE EDITOR

- ▶ 1. Go to the list of triggers Administration > Workflows
- ▶ 2. Enable the trigger
- ▶ 3. Edit the trigger you want to create a workflow for
- ▶ 4. Drag an action module from the side panel to the canvas
- ▶ 5. Drag another action module or a logic module from the side panel to the canvas
- ▶ 6. From the trigger output, drag an arrow into the action's input (left side)
- ▶ 7. Continue linking modules with the input/output system until all wanted modules are connected
- ▶ 8. Find an action that would execute the desired trigger
- ▶ 9. Execute the action and observe the effect!
- ▶ 10. Optionally, enable debug mode to see realtime execution
- ▶ 10.1. Even more text to make the slide even more unreadable
- ▶ 10.2. And even more boring

CREATING A WORKFLOW WITH THE EDITOR

- ▶ 1. Go to the list of triggers Administration > Workflows
- ▶ 2. Enable trigger
- ▶ 3. Edit the trigger
- ▶ 4. Drag and drop the trigger to the canvas
- ▶ 5. Drag and drop the action from the side
- ▶ 6. From the action menu, select the action
- ▶ 7. Continue to configure the action until all variables are set
- ▶ 8. Find an action to trigger the workflow
- ▶ 9. Execute the workflow
- ▶ 10. Option 10.1. Even more boring
- ▶ 10.2. And even more boring



Writing
more
slides

Doing
another
live demo

w for

e

t system

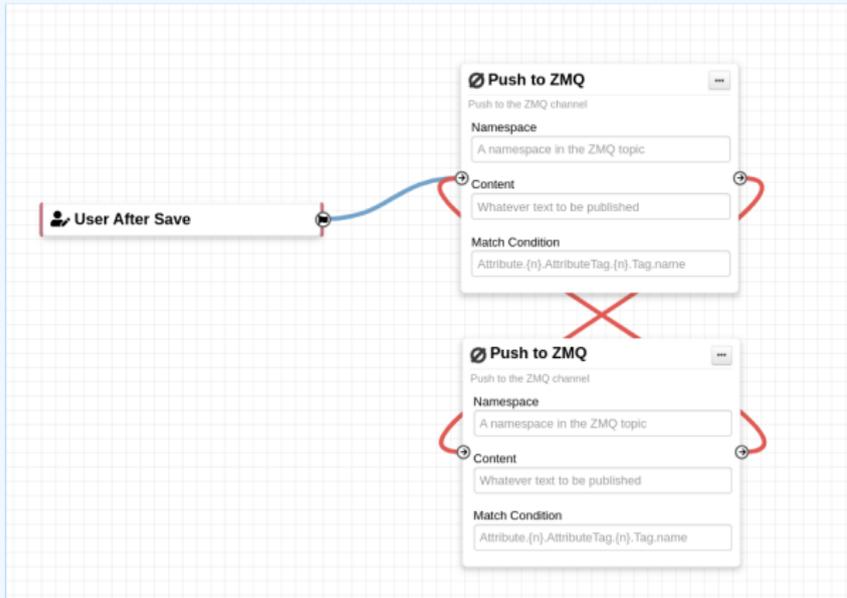
trigger

e execution
unreadable

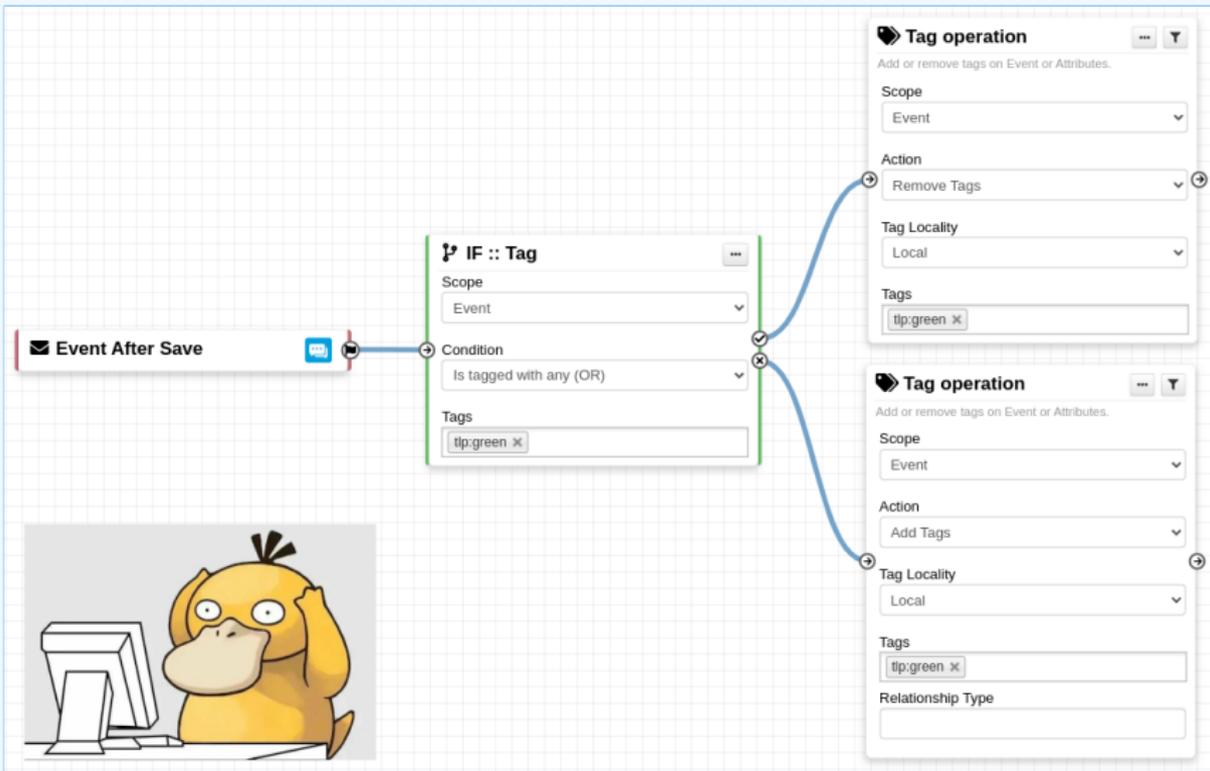
CONSIDERATIONS WHEN WORKING WITH WORKFLOWS

WORKING WITH THE EDITOR - OPERATIONS NOT ALLOWED

Execution loop are not authorized



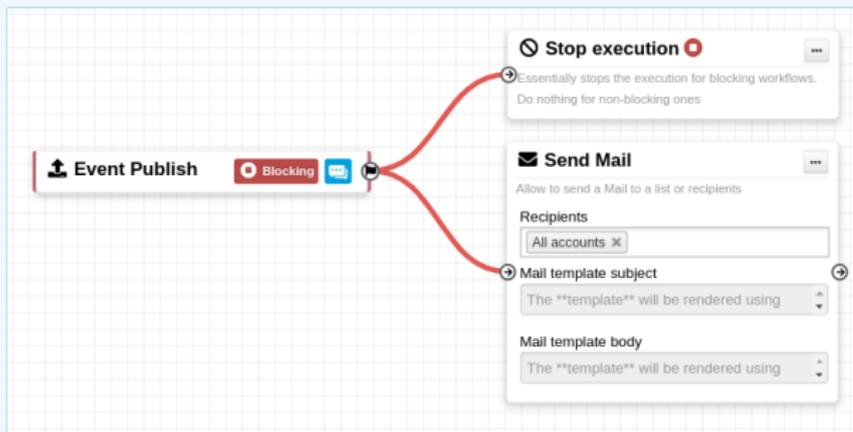
RECURSIVE WORKFLOWS



⚠ **Recursion:** If an action re-run the workflow

WORKING WITH THE EDITOR - OPERATIONS NOT ALLOWED

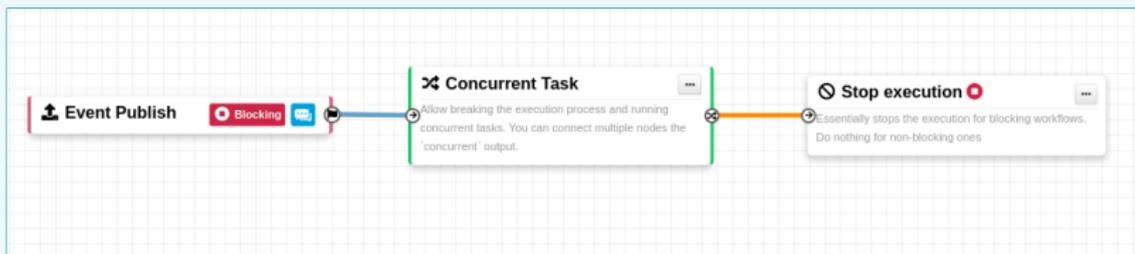
Multiple connections from the same output



- Execution order not guaranteed
- Confusing for users

Cases showing a warning:

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



ADVANCED USAGE



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: ✕

Blueprint Content: **1 node**

 1

Webhook module pre-configured for debugging purposes

Blueprints sources:

1. Created or imported by users
2. From the [MISP/misp-workflow-blueprints](https://github.com/MISP/misp-workflow-blueprints) repository¹

¹<https://github.com/MISP/misp-workflow-blueprints>

HASH PATH FILTERING

Filtering and checking conditions using hash path expression.

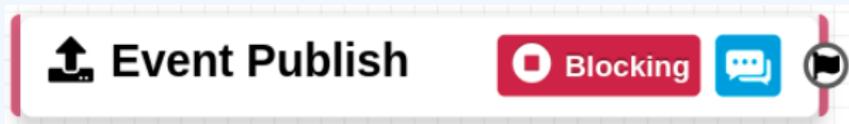
```
1 $path_expression = '{n}[name=fred].id';
2 $users = [
3   {'id': 123, 'name': 'fred', 'surname': 'bloggs'},
4   {'id': 245, 'name': 'fred', 'surname': 'smith'},
5   {'id': 356, 'name': 'joe', 'surname': 'smith'},
6 ];
7 $ids = Hash::extract($users, $path_expression);
8 // => $ids will be [123, 245]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

```
{
  "Attribute": [
    {
      "type": "domain",
      "value": "cti-summit.org",
      "Tag": [
        {
          "name": "t1p:red",
          "colour": "#CC0033"
        }
      ]
    }
  ]
}
```

The screenshot shows a configuration window titled "IF :: Generic". It contains three fields:

- Value:** t1p:red
- Operator:** In
- Hash path:** Attribute.{n}.Tag

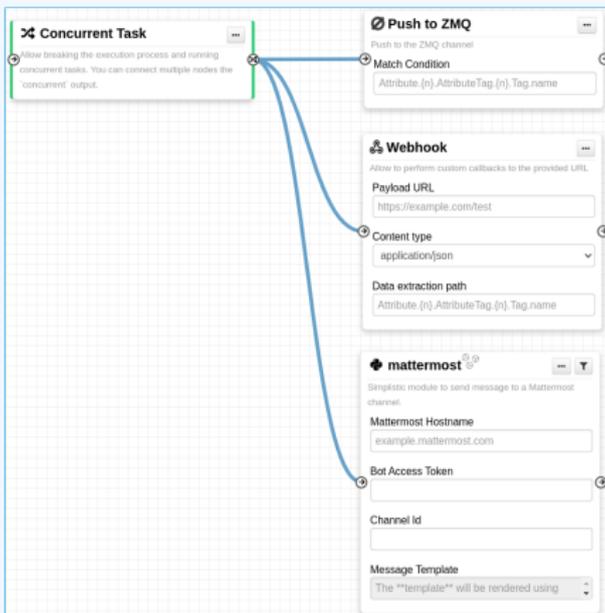
There are navigation icons on the right side of the form: a left arrow, a checkmark, and a close button (X).



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

LOGIC MODULE: CONCURRENT TASK

- Logic module allowing **multiple output** connections
- **Postpone the execution** for remaining modules
- Blocking modules  **cannot cancel** ongoing operations 



DEBUGGING

DEBUGGING WORKFLOWS: LOG ENTRIES

- Workflow execution is logged in the application logs:
 - ▶ `/admin/logs/index`
- Or stored on disk in the following file:
 - ▶ `/app/tmp/logs/workflow-execution.log`

Logs

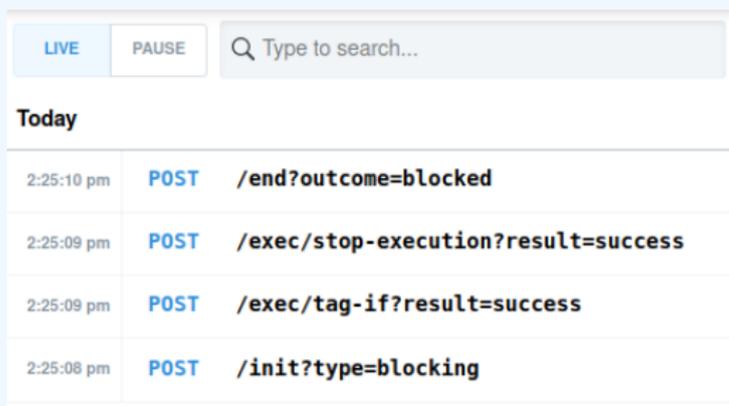
« previous next »

Emails Authentication issues MISIP Update results Setting changes Warnings and errors

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  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



Today		
2:25:10 pm	POST	<code>/end?outcome=blocked</code>
2:25:09 pm	POST	<code>/exec/stop-execution?result=success</code>
2:25:09 pm	POST	<code>/exec/tag-if?result=success</code>
2:25:08 pm	POST	<code>/init?type=blocking</code>

■ Test custom modules with custom input

Stateless module execution

Module parameters

Payload URL

Content type

Data extraction path

Input data

Convert input data into MISP core format

Module Input Data

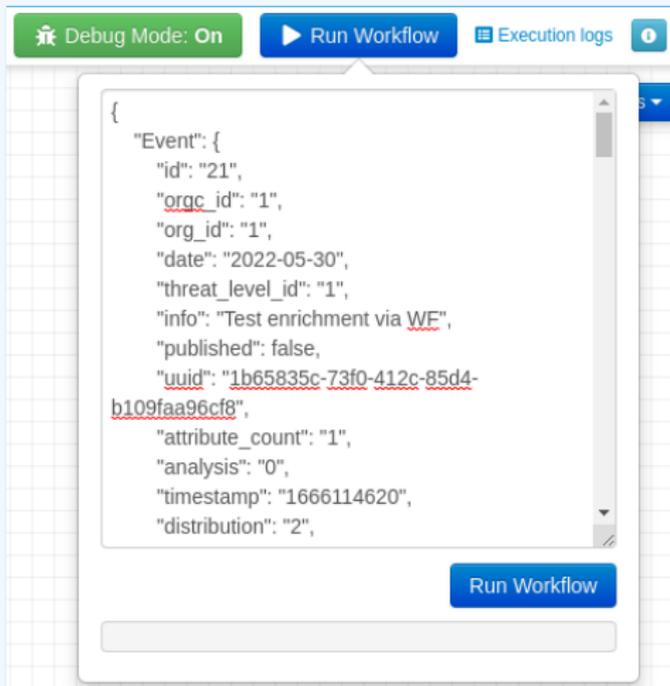
```
{  
  "foo": "bar"  
}
```

Execute module

Execution result: 200 [56 ms]

DEBUGGING MODULES: RE-RUNNING WORKFLOWS

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



The screenshot shows a workflow debugging interface. At the top, there is a green button labeled "Debug Mode: On" with a bug icon, a blue "Run Workflow" button with a play icon, and a link to "Execution logs" with a document icon. Below this is a large text area containing a JSON object representing an event. The JSON is as follows:

```
{
  "Event": {
    "id": "21",
    "orgc_id": "1",
    "org_id": "1",
    "date": "2022-05-30",
    "threat_level_id": "1",
    "info": "Test enrichment via WF",
    "published": false,
    "uuid": "1b65835c-73f0-412c-85d4-b109faa96cf8",
    "attribute_count": "1",
    "analysis": "0",
    "timestamp": "1666114620",
    "distribution": "2",
  }
}
```

At the bottom right of the text area, there is another blue "Run Workflow" button. Below the text area is a light gray input field.

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_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    }
24 }
```

- `app/Lib/WorkflowModules/action/[module_name].php`
- Designed to be easily extended
 - ▶ Helper functions
 - ▶ Module configuration as variables
 - ▶ Implement runtime logic

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
39
40
41 > def introspection():-
```

- Similar to how other misp-modules are implemented
 - ▶ Helper functions
 - ▶ Module configuration as variables
 - ▶ Implement runtime logic

- Chat notification a community when new user joins an instance
- Trigger on any action via log entries
- Extend existing MISP behavior: Push correlation in another system
- Sanity check to block publishing
- ...

FUTURE WORKS

- More 🎬 modules
- More ➡ modules
- More 🦊 triggers
- More documentation
- Recursion prevention system
- On-the-fly data override?



FINAL WORDS

- Feature designed to quickly and cheaply support CTI pipeline
- **Beta:** Feature unlikely to change. But still..
- Waiting for feedback!

