Automation with Workflows in MISP

Advanced version

Sami Mokaddem

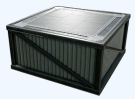
MISP Project
https://www.misp-project.org/

Example of use-cases

- **Notification** on specifc 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
  - Curation pipelines
- **Hook** capabilities
  - Assign tasks and notify incident response team members
- ...

Workflow - Fundamentals

Objective: Start with the foundation to understand the basics



Triggers

Currently 11 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 »

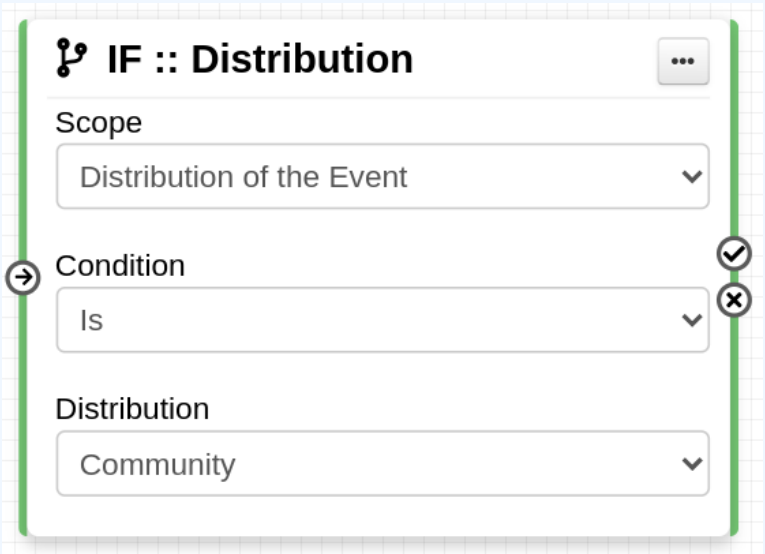
All | attribute | event | log | object | others | post | user | Blocking | Enabled | Disabled

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	110	✗	✓	160	2023-09-14 06:54:37	☐	✗	
Enrichment Before Query	others	low	2226	✓	✓	162	2023-10-09 07:56:42	☐	✓	
Event After Save	event	high	191	✗	✓	175	2023-10-02 14:55:19	☐	✗	
Event After Save New	event	low	7	✗	✓	182	2023-03-16 14:05:07	☐	✗	
Event After Save New From Pull	event	low	6	✗	✓	183	2023-10-09 07:57:02	☐	✗	
Event Publish	event	low	2	✓	✓	188	2023-10-09 07:56:25	☐	✓	
Log After Save	log	high	0	✗	✗	185	2023-06-05 13:26:50	☐	✗	
Object After Save	object	high	35	✗	✓	161	2023-06-05 13:27:00	☐	✗	
Post After Save	post	low	36	✗	✗	176	2022-07-28 13:59:51	☐	✗	
User After Save	user	low	0	✗	✗	181	2022-08-05 07:19:46	☐	✗	
User Before Save	user	low	42	✓	✗	158	2023-06-05 13:27:25	☐	✗	

Page 1 of 1, showing 1 records out of 11 total, starting on record 1, ending on 11

Logic modules allow to redirect the execution flow

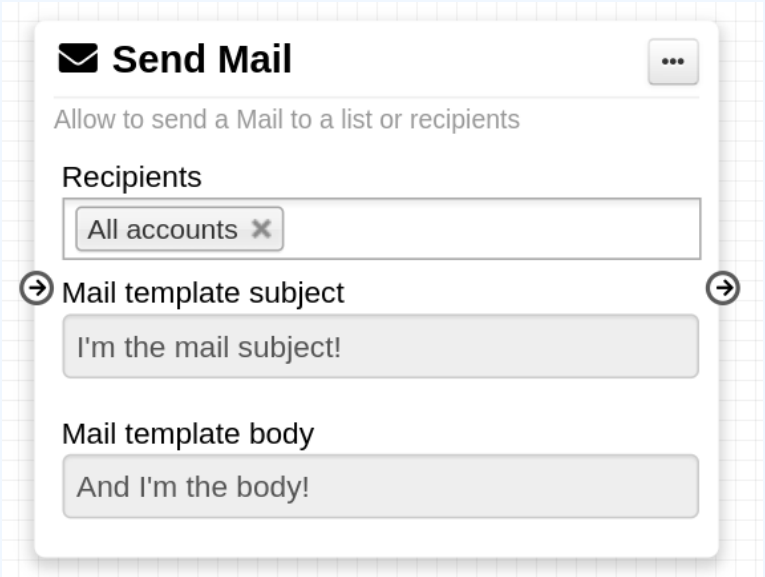
- A MISP Event is tagged with `tlp:red`
- The distribution of an Attribute is a sharing group
- The creator organisation is `circl.lu`
- Or any other **generic** conditions



Actions modules

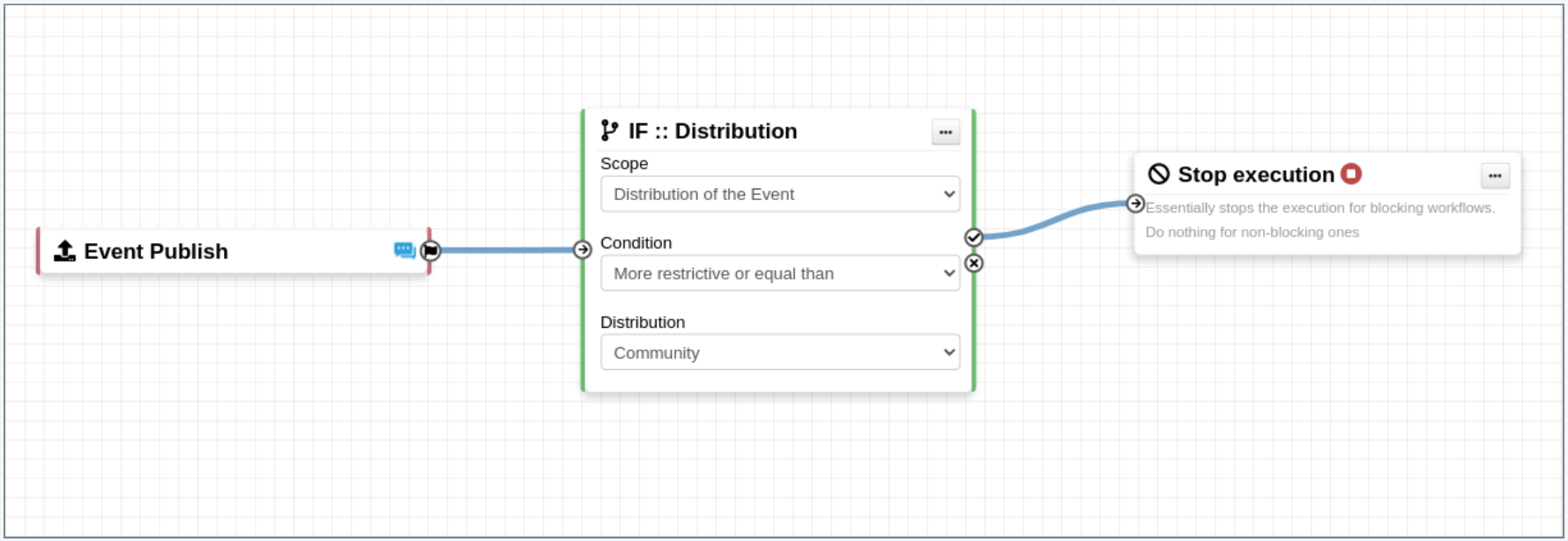
Action modules allow to executes operations

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



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

Built-in **default** modules

- Part of the MISP codebase
- Ready to use once enabled

User-defined **custom** modules

Written in PHP

- Extend existing modules
- MISP code reuse

Written in Python

- Can rely on extensive python libraries
- Easier to write
- Rely on the **enrichment service** misp-modules

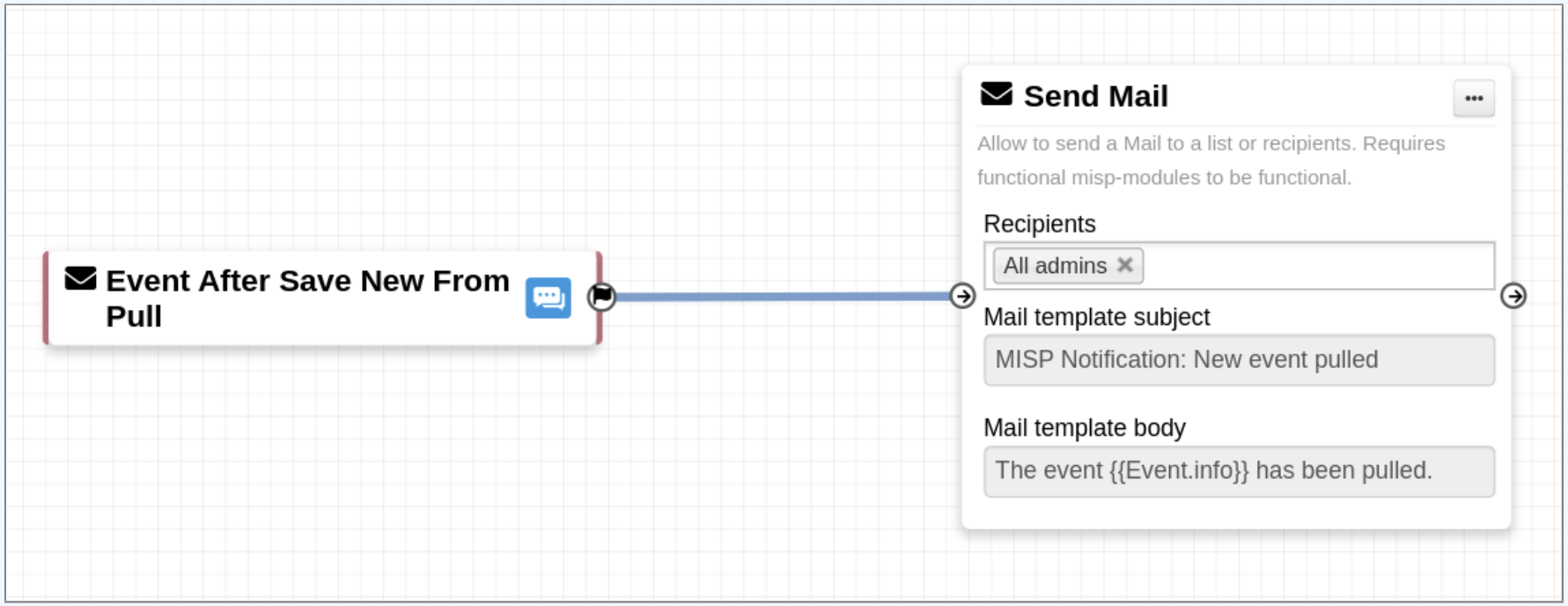
Demo by examples

WF-1. Send an email to **all admins** 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

Demo WF-1: Send an email to **ALL ADMINS** when a new event has been pulled



Demo 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

Creating a workflow with the editor

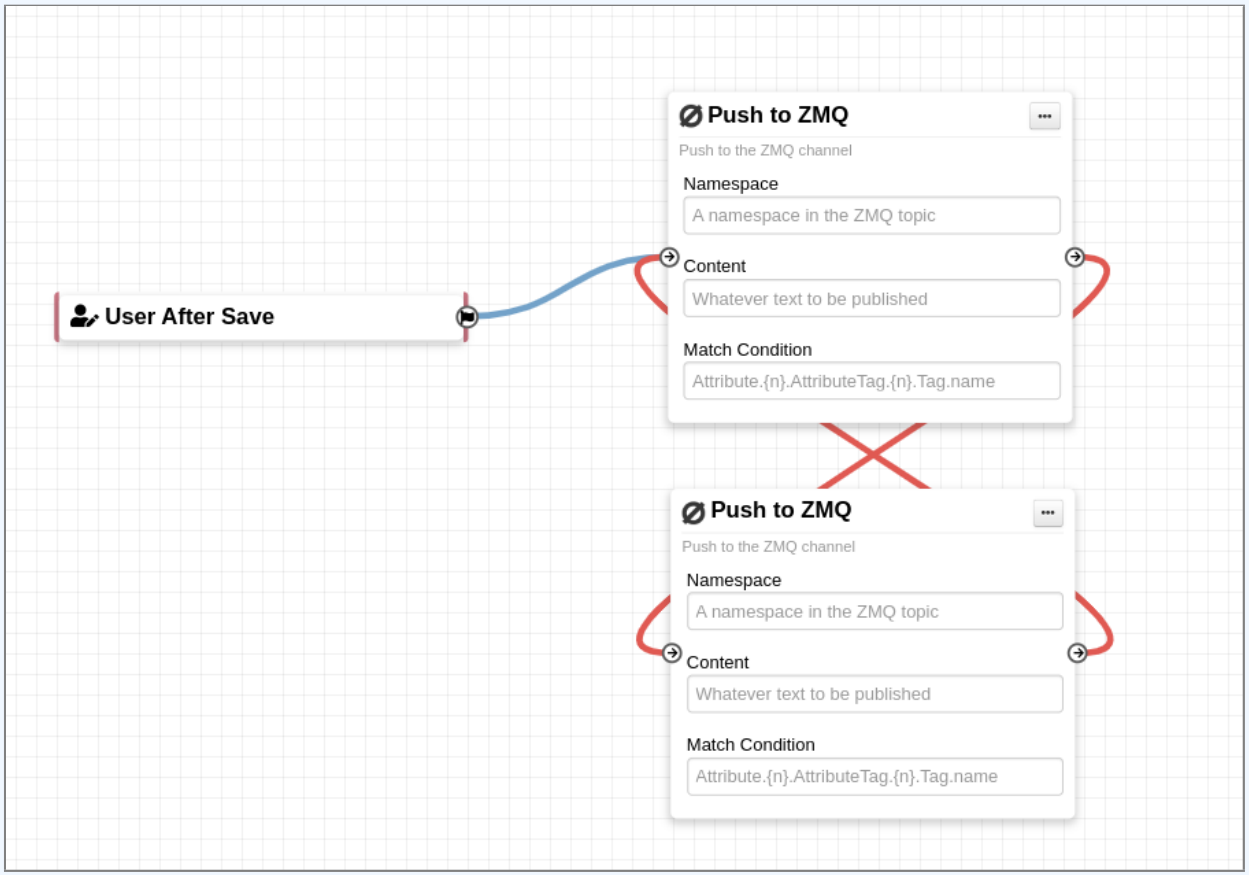
1. Prevent event publication **if tlp:red** tag
   - Send a mail to `admin@admin.test` about potential data leak
2. **else**, send a notification on Mattermost

Considerations when working with workflows

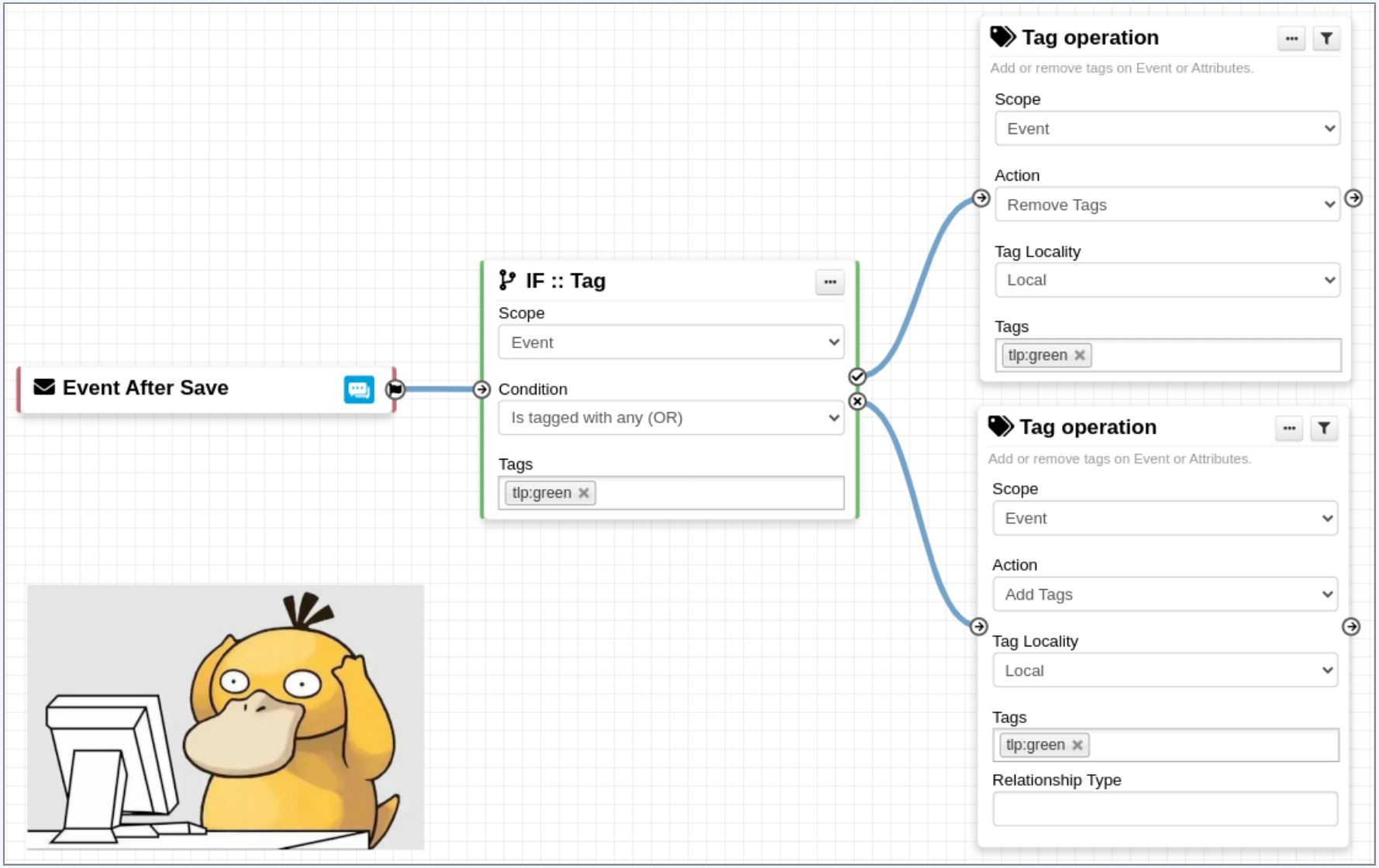
Objective: Overview of some common pitfalls

Working with the editor - Operations not allowed

Execution loop are not authorized



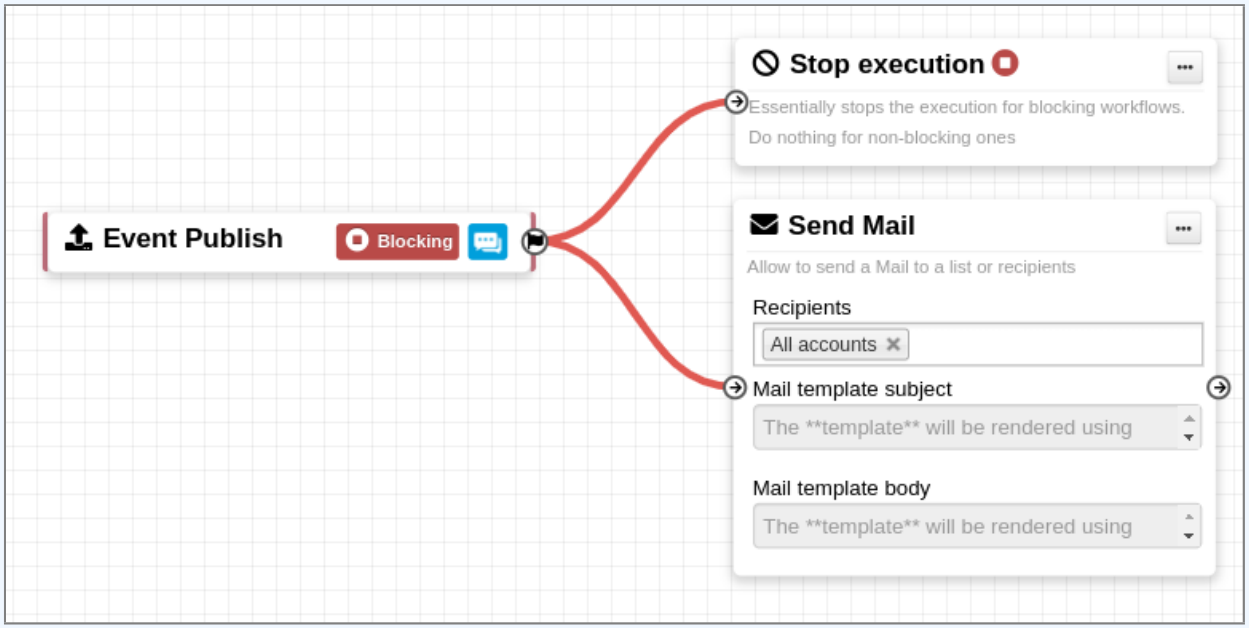
Recursive workflows



⚠Recursion: If an action re-run the workflow

Working with the editor - Operations not allowed

Multiple connections from the same output



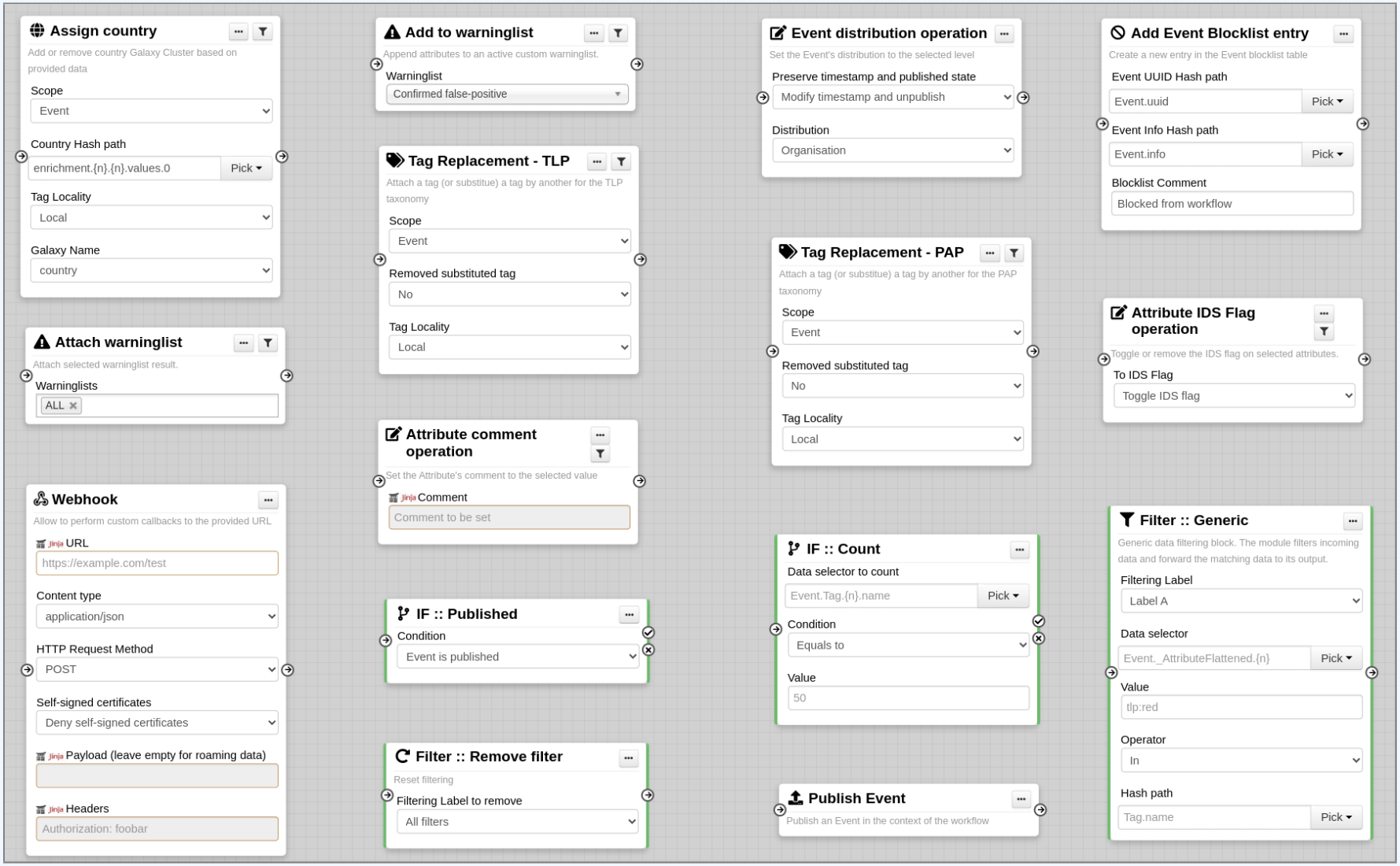
- Execution order not guaranted
- Confusing for users

New recent features

New recent features I

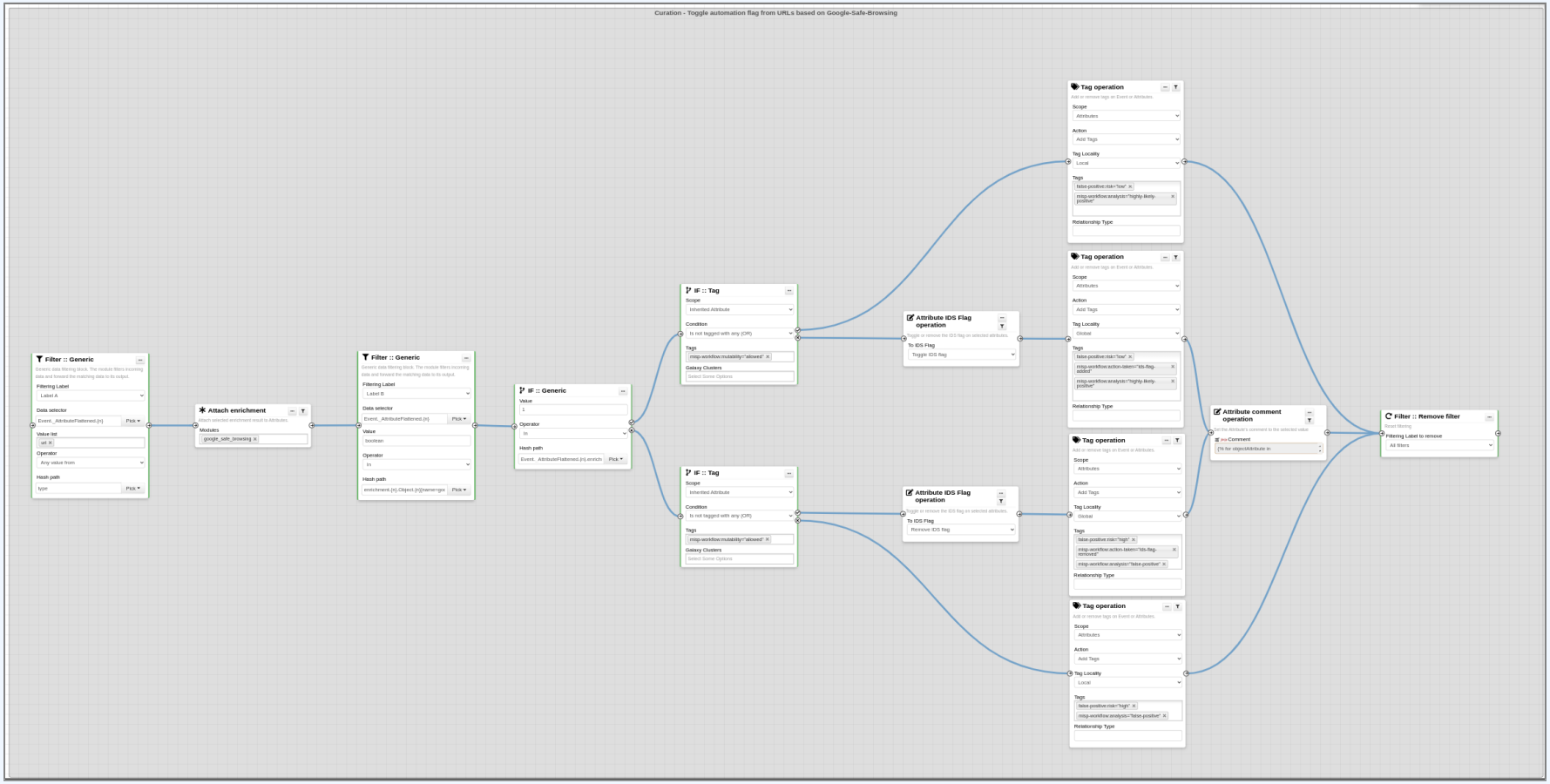
- New action modules & improvements
  - `Assign country`
  - `Attach warninglist`
  - `Attribute operations`
  - `Tag replacements`
  - `Webhook`, ⋯
- New logic modules & improvements
  - `Filter :: Generic`
  - `Filter :: Remove`
  - `IF :: *`

New recent features I



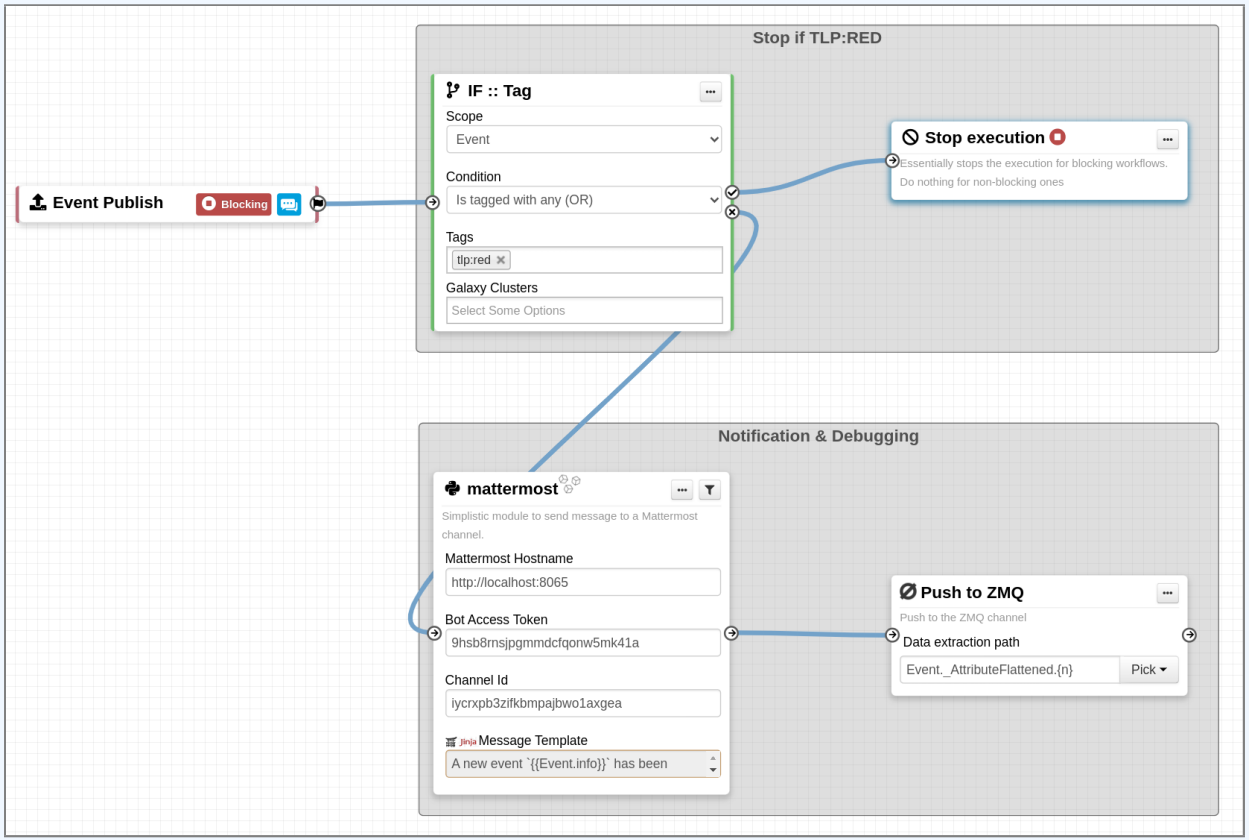
New recent features II

~ 12 New blueprints for IoC curation



New recent features III

- UI improvements
  - Frame to annotate and group modules
  - More documentation (Format, Jinja2 syntax)
  - Collapsible sidebar and quick node insert
  - Hash path picker



Advanced usage

Advanced usage

Objective:

- Blocking workflows
- Blueprints
- Filtering
- Data format
- Debugging

Blocking and non-blocking

Two types of workflows:

Blocking Workflows

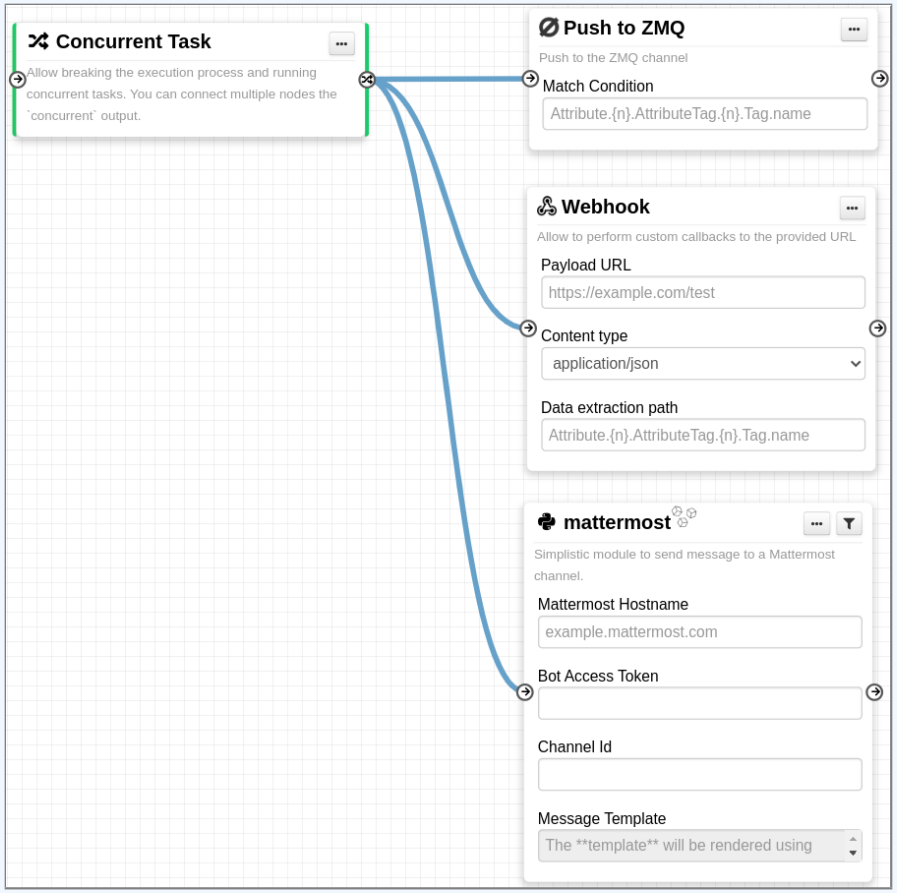
- Can prevent / block the original event to happen
- If a **blocking module** blocks the action
- `event-publish`, `event-before-save`, `enrichment-before-query`, ⋯

Non blocking Workflows execution outcome has no impact

- No way to prevent something that happened in the past
- `event-after-save`, `attribute-after-save` `log-after-save`, ⋯

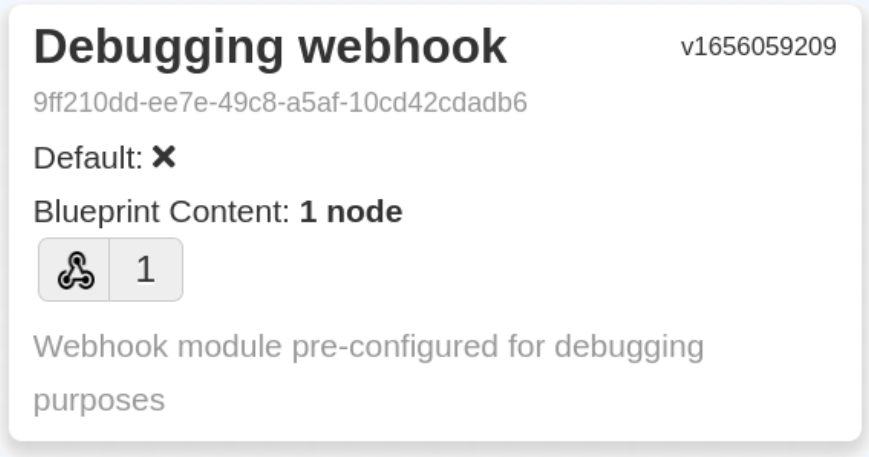
Logic module: Concurrent Task

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



Workflow blueprints

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



Blueprints sources: `MISP/misp-workflow-blueprints` repository[1]

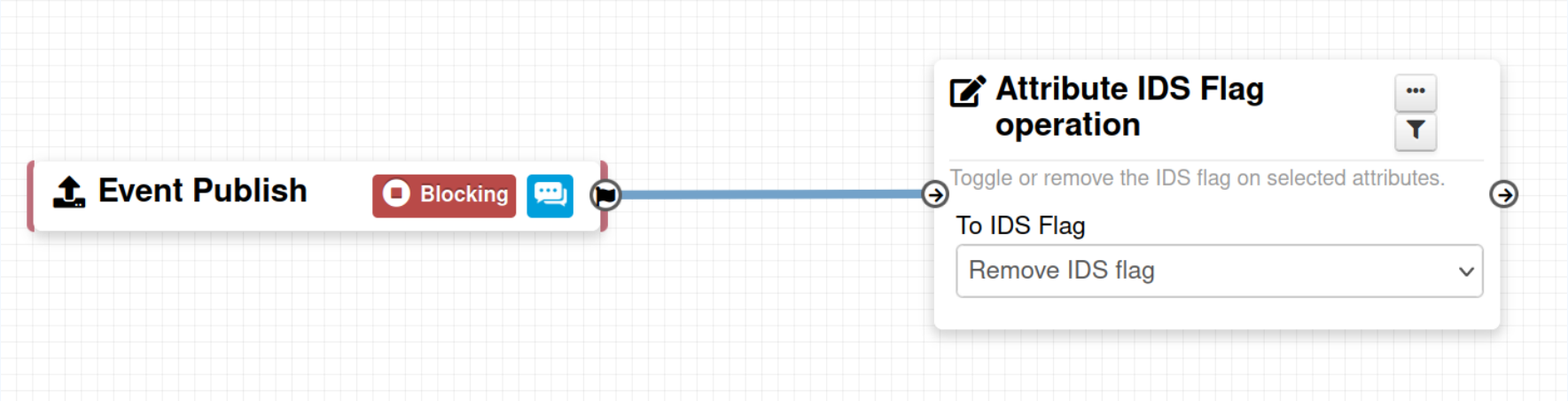
- Block actions if any attributes have the `PAP:RED` or `tlp:red` tag
- Curation pipeline
- Enrich data from 3rd-party

[1] https://github.com/MISP/misp-workflow-blueprints

Filtering

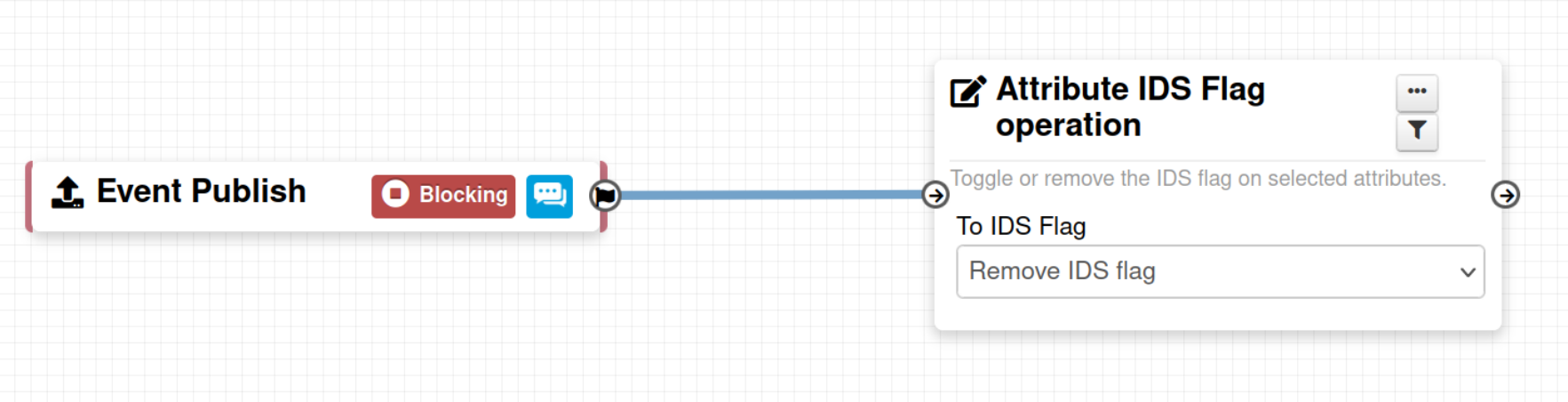
Fitlering data on which to apply a module

What is the outcome of executing this workflow?



Fitlering data on which to apply a module

What is the outcome of executing this workflow?

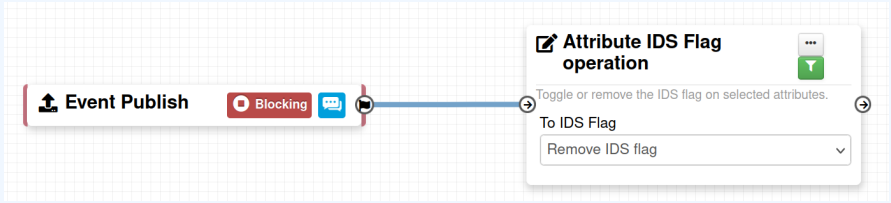


All Attributes get their `to_ids` turned off.

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

→ Hash path filtering!

Fitlering data on which to apply a module



Node Filtering

Element selector

Event._AttributeFlattened.{n}

Value

comment

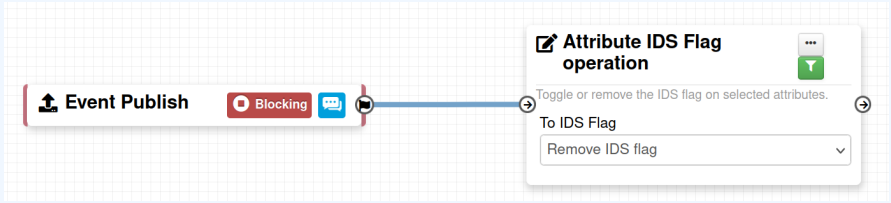
Operator

In

Hash Path

type

Fitlering data on which to apply a module



Node Filtering

Select elements on which to apply the filtering

Element selector

Event._AttributeFlattened.{n}

Value

Fixed value

comment

Operator

In

Comparison operator

Hash Path

type

Data point to get the value

Fitlering data on which to apply a module


```
{
    "Event": {
        "id": "64",
        "orgc_id": "1",
        "org_id": "1",
        "date": "2023-05-03",
        "info": "Core format sample",
        "published": false,
        "uuid": "b9557473-bb46-4c65-b69e-974b3c93c1f4",
        "analysis": "0",
        "timestamp": "1683117902",
        "Attribute": [
            {
                "id": "1695",
                "type": "ip-src",
                "category": "Network activity",
                "to_ids": true,
                "uuid": "9ac36927-d874-4094-bf4c-f922c1e9cc35",
                "event_id": "64",
                "distribution": "5",
                "timestamp": "1683117902",
                "comment": "",
                "sharing_group_id": "0",
                "deleted": false,
                "disable_correlation": false,
                "object_id": "0",
                "object_relation": null,
                "first_seen": null,
                "last_seen": null,
                "value": "8.8.8.8",
                "Galaxy": [],
                "ShadowAttribute": [],
                "Tag": [
                    {
                        "id": "137",
                        "name": "PAP:AMBER",
                        "colour": "#ffa800",
```


Extract elements on which to apply the filter

{"**id**": 1, "**value**": "8.8.8.8", "**type**": "ip-src"}

{"**id**": 2, "**value**": "8.8.4.4", "**type**": "ip-dst"}

{"**id**": 3, "**value**": "circl.lu", "**type**": "domain"}

{"**id**": 4, "**value**": "-- test --", "**type**": "comment"}

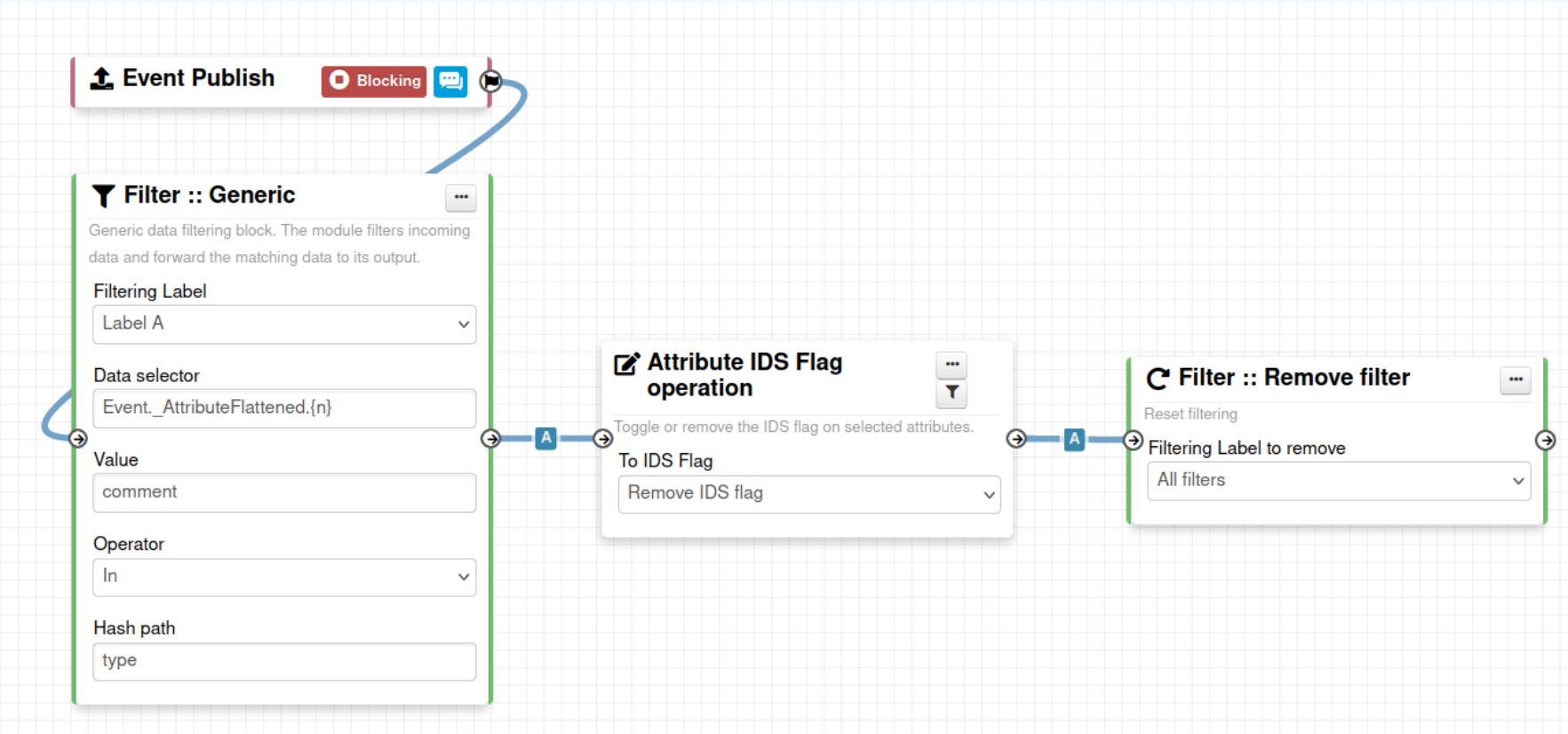
For each element

1. Access the entry's value from the hash path
2. Compare two values using the operator
3. Discard invalid elements

{"**id**": 4, "**value**": "-- test --", "**type**": "comment"}

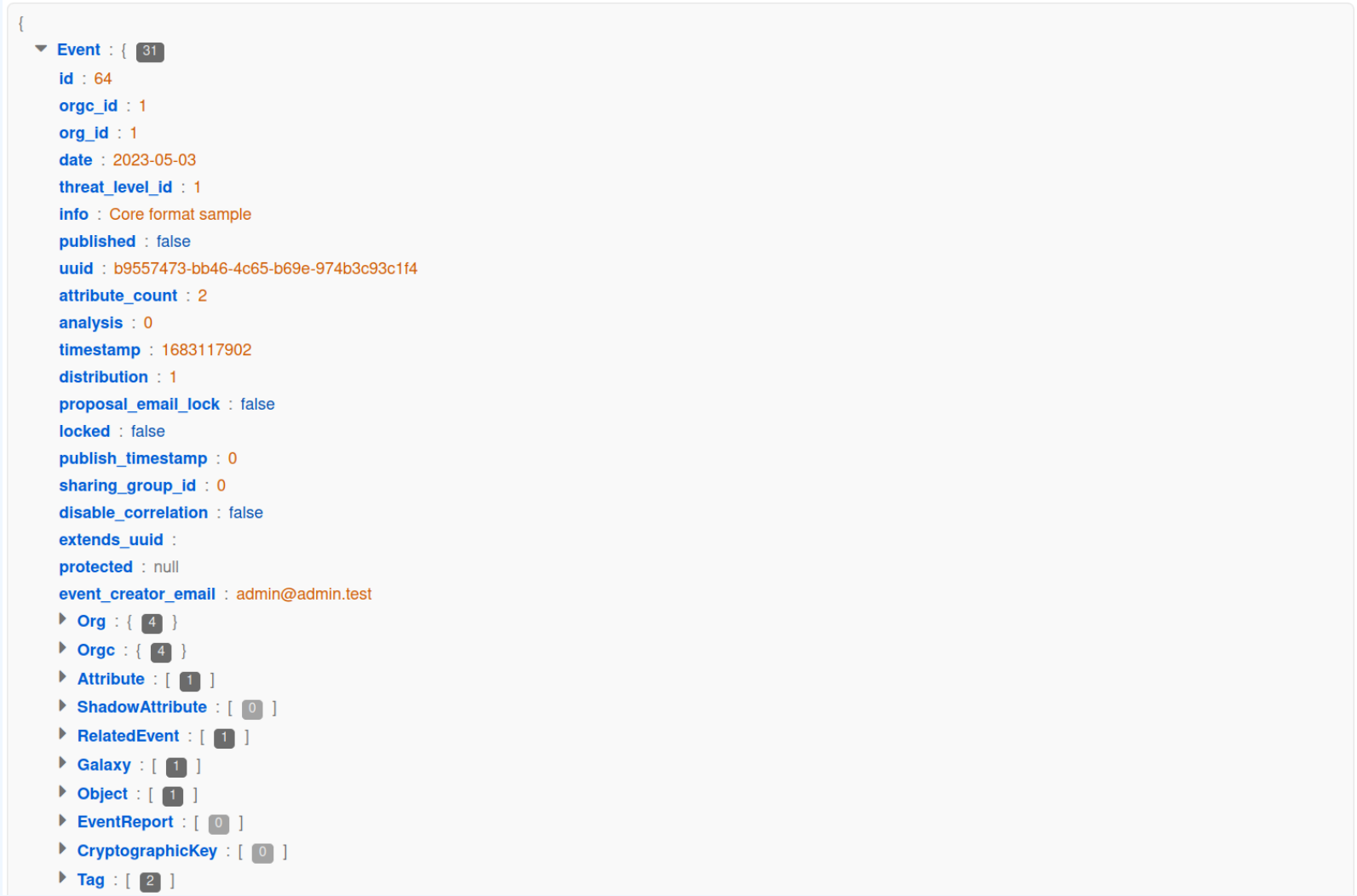
Fitlering data on which to apply on multiple modules

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



Data format in Workflows

- In most cases, the format is the **MISP Core format**
  - Attributes are **always encapsulated** in the Event or Object



Hash path filtering - Example

```
{
    "Event": {
        "uuid": ...
        "timestamp": ...
        "distribution": 1,
        "published": false,
        "Attribute": [
            {
                "type": "ip-src",
                "value": "8.8.8.8", ...
            },
            {
                "type": "domain",
                "value": "misp-project.org", ...
            }
        ],
        ...
    }
}
```

1. Access Event distribution
    - `Event.distribution`

Hash path filtering - Exercise (1)

```
{
    "Event": {
        "uuid": ...
        "distribution": 1,
        "published": false,
        "Attribute": [
            {
                "type": "ip-src",
                "value": "8.8.8.8", ...
            },
            {
                "type": "domain",
                "value": "misp-project.org", ...
            }
        ],
        ...
    }
}
```

2. Access Event published state

Hash path filtering - Exercise (1)

```
{
    "Event": {
        "uuid": ...
        "distribution": 1,
        "published": false,
        "Attribute": [
            {
                "type": "ip-src",
                "value": "8.8.8.8", ...
            },
            {
                "type": "domain",
                "value": "misp-project.org", ...
            }
        ],
        ...
    }
}
```

2. Access Event published state
    - `Event.published`

Hash path filtering - Exercise (2)

```
{
    "Event": {
        "uuid": ...
        "distribution": 1,
        "published": false,
        "Attribute": [
            {
                "type": "ip-src",
                "value": "8.8.8.8", ...
            },
            {
                "type": "domain",
                "value": "misp-project.org", ...
            }
        ],
        ...
    }
}
```

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

Hash path filtering - Exercise (2)

```
{
    "Event": {
        "uuid": ...
        "distribution": 1,
        "published": false,
        "Attribute": [
            {
                "type": "ip-src",
                "value": "8.8.8.8", ...
            },
            {
                "type": "domain",
                "value": "misp-project.org", ...
            }
        ],
        ...
    }
}
```

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

Hash path filtering - Exercise (3)

```
{
    "Event": {
        "Attribute": [
            {
                "type": "ip-src",
                "value": "8.8.8.8",
                "Tag": [
                    {
                        "name": "PAP:AMBER", ...
                    }
                ], ...
            }
        ],
        ...
    }
}
```

3. Access all Tags attached to Attributes

Hash path filtering - Exercise (3)

```
{
    "Event": {
        "Attribute": [
            {
                "type": "ip-src",
                "value": "8.8.8.8",
                "Tag": [
                    {
                        "name": "PAP:AMBER", ...
                    }
                ], ...
            }
        ],
        ...
    }
}
```

3. Access all Tags attached to Attributes
   - `Event.Attribute.{n}.Tag.{n}.name`

Hash path filtering - Exercise (4)

```
{
    "Event": {
        "Tag": [
            {
                "name": "tlp:green", ...
            }
        ], ...
        "Attribute": [
            {
                "value": "8.8.8.8",
                "Tag": [
                    {
                        "name": "PAP:AMBER", ...
                    }
                ], ...
            }
        ],
    }
}
```

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

Hash path filtering - Exercise (4)

```
{
    "Event": {
        "Tag": [
            {
                "name": "tlp:green", ...
            }
        ], ...
        "Attribute": [
            {
                "value": "8.8.8.8",
                "Tag": [
                    {
                        "name": "PAP:AMBER", ...
                    }
                ], ...
            }
        ],
    }
}
```

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

Hash path filtering - Exercise (4)

```
{
    "Event": {
        "Tag": [...],
        "Attribute": [
            {
                "value": "8.8.8.8",
                "_allTags": [
                    {
                        "name": "tlp:green",
                        "inherited": true, ...
                    },
                    {
                        "name": "PAP:AMBER",
                        "inherited": false, ...
                    }
                ],
            }
        ...
}
```

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

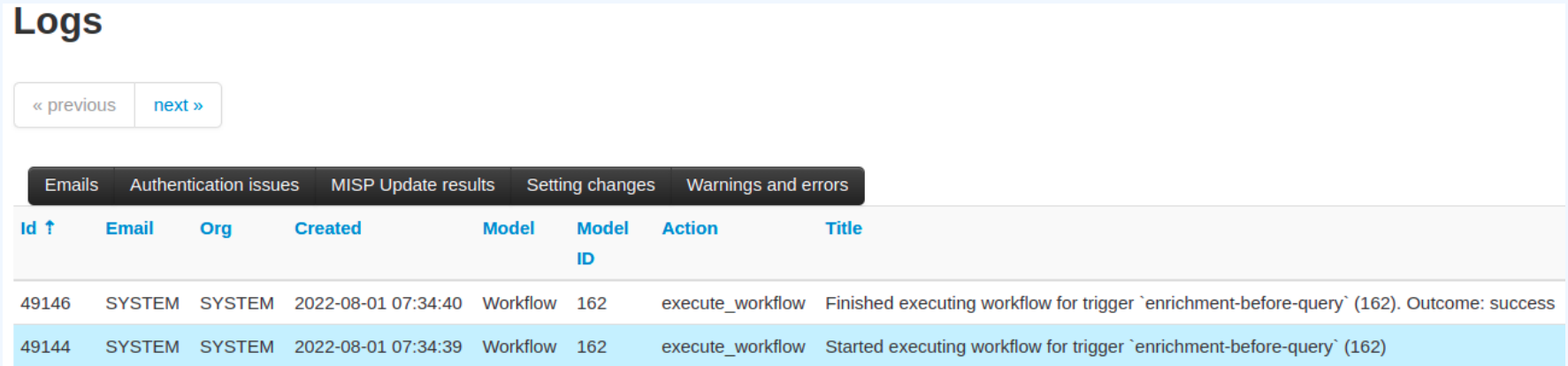
Data format in Workflows

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

Debugging

Debugging Workflows: Log Entries

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



Logs

« previous | next »

Emails | Authentication issues | MISP 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)

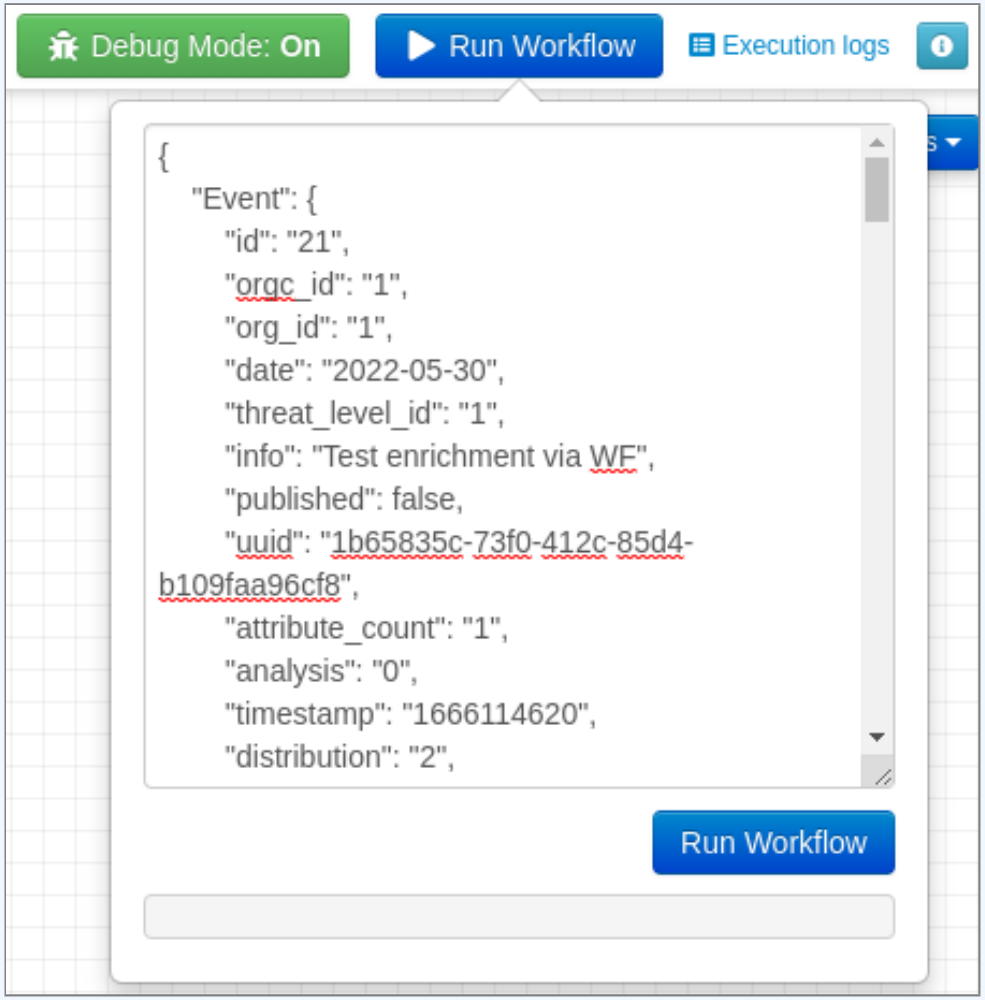
Debugging Workflows: Debug mode

- 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

Debugging modules: Re-running workflows

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



Debugging options

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

Should I migrate to MISP Workflows

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

- I have a curation pipeline using the API, should I port it to workflows?
  - **No** in general, but WF can be used to start the curation process or perform simple pre-processing
- What if I want to **block** some actions
  - Put the blocking logic in the WF, keep the remaining outside
- Bottom line is **Keep it simple** for you to maintain

Future works

- More action modules 
- More logic modules 
- More triggers 
- Recursion prevention system
- Improvement for logging

Final words

- Designed to **quickly** and **cheaply** integrate MISP in CTI pipelines
- Waiting for feedback!
  - New triggers?
  - New modules?