An Introduction to Workflows in MISP

MISP - Threat Sharing

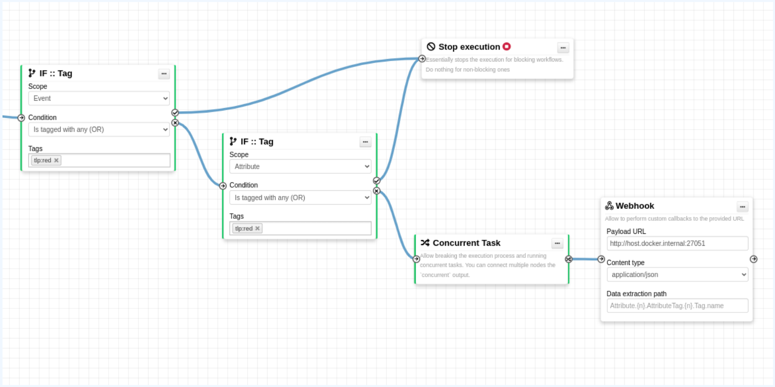
CIRCL / Team MISP Project

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

NSPA

Content of the presentation

- MISP Workflows fundamentals
- Getting started
- Design of the system & how it can be extended



IF :: Tag
Scope: Event
Condition: Is tagged with any (OR)
Tags: tlp:red

IF :: Tag
Scope: Attribute
Condition: Is tagged with any (OR)
Tags: tlp:red

Stop execution
Essentially stops the execution for blocking workflows. Do nothing for non-blocking ones

Concurrent Task
Allow breaking the execution process and running concurrent tasks. You can connect multiple nodes the `concurrent` output.

Webhook
Allow to perform custom callbacks to the provided URL
Payload URL: http://host.docker.internal:27051
Content type: application/json
Data extraction path: Attribute.{n}.AttributeTag.{n}.Tag.name

What problems are we trying to tackle

- Initial idea came from GeekWeek7.5[1]

- Needs:
  - Prevent default MISP behaviors
  - Hook specific actions via callbacks
- Use-cases:
  - Prevent publication of events not meeting some criterias
  - Prevent querying thrid-party services (e.g. virustotal) with sensitive information
  - Send a notification in a chat room
  - And much much more..

[1]Workshop organized by the Canadian Cyber Center

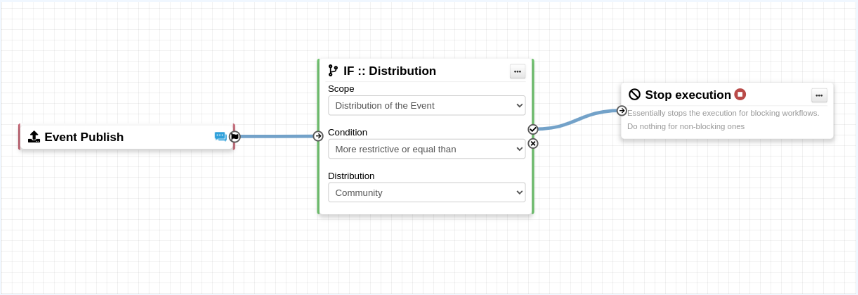
Workflow - Fundamentals

Simplistic overview of a Workflow in action

1. An **action** is performed in MISP
2. If there is an **enabled** Workflow for that **action**, run it
3. If all went fine, MISP **continue** to perform the action
    - The operation can potentially be cancelled by `blocking` modules

Terminology

- **workflow**: Sequence of all operations (nodes) to be executed. Basically the whole graph
- **execution path**: A path composed of nodes
- **trigger**: Starting point of a workflow. Triggers are called when specific actions happen in MISP
  - A trigger can only have one workflow and vice-versa



Scope: Distribution of the Event
Condition: More restrictive or equal than
Distribution: Community

Essentially stops the execution for blocking workflows. Do nothing for non-blocking ones

Workflow execution process

Typical execution process:

1. An action happens in MISP
2. The workflow associated to the trigger is ran
3. Execution result?
   - `success`: Proceed the action
   - `failure` | `blocked`: Cancel the action

Example for Event publish:

1. An Event is about to be published
2. MISP executes the workflow listening to the `event-publish` trigger
   - **success**: Proceed the publishing action
   - **failure** | `blocked`: Stop publishing and log the reason

Blocking and non-blocking Workflows

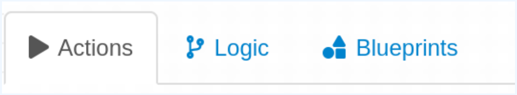
Currently 2 types of workflows:

- **Blocking**: Completion of the initial action can be prevented
  - If a **blocking module** blocks the action
  - If a **blocking module** raises an exception
- **Non-blocking**: Workflow execution outcome has no impact
  - **Blocking modules** become **non-blocking**
  - Execution proceed unless on exception

Execution context

- Workflows can be triggered by **any users**
- Workflows can be triggered by actions done via the **UI** or **API**
- However, the user for which the workflow executes has:
  - The `site-admin` permission
  - Is from the `MISP.host_org_id`
- Ensures data is processed regardless of ownership and access: **no ACL**

Classes of Workflow modules



3 classes of modules

- **action**: Allow to executes actions, callbacks or scripts
  - Can stop execution
  - e.g. Webhook, block the execution, perform enrichments, ...
- **logic**: Allow to redirect the execution flow.
  - IF condition, fork the blocking execution into a non-blocking one, ...
- **blueprint**: Allow to reuse composition of modules
  - Can save subworkflows and their module's configuration

Sources of Workflow modules

3 sources of action modules

- Built-in **default** modules
  - Written in PHP
  - Can use MISP's built-in functionalities (restsearch, enrichment, push to zmq, ...)
  - Fast and easier to interact with for those having internal knowledge of MISP
  - `app/Model/WorkflowModules/action/[module_name].php`
- User-defined **custom** modules
  - Can extend existing default modules
  - `app/Lib/WorkflowModules/action/[module_name].php`

Sources of Workflow modules

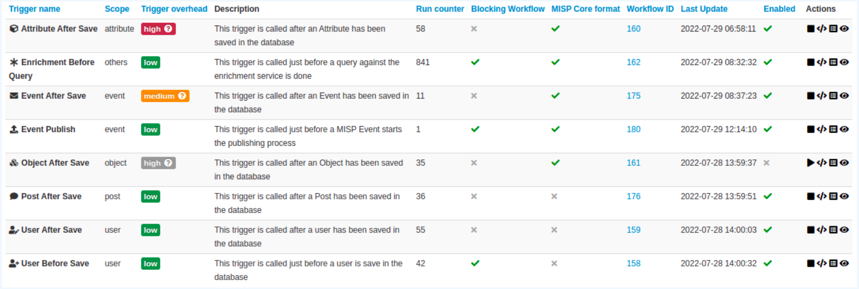
3 sources of action modules

- Modules from the **enrichment service**
  - **Default** and **custom** modules
  - `From the misp-module service`
  - Written in Python
  - Can use any python libraries
  - New `misp-module` module type: `action`

→ Both the PHP and Python systems are **plug-and-play**

Triggers currently available

Currently 8 triggers can be hooked. 3 being **blocking**.



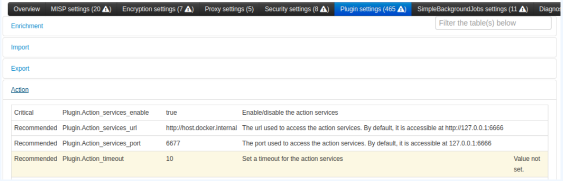
Trigger name	Scope	Trigger overhead	Description	Run counter	Blocking Workflow	MISP Core format	Workflow ID	Last Update	Enabled	Actions
Attribute After Save	attribute	high ?	This trigger is called after an Attribute has been saved in the database	58	✗	✓	160	2022-07-29 06:58:11	✓	■ </> ▤ ◉
Enrichment Before Query	others	low	This trigger is called just before a query against the enrichment service is done	841	✓	✓	162	2022-07-29 08:32:32	✓	■ </> ▤ ◉
Event After Save	event	medium ?	This trigger is called after an Event has been saved in the database	11	✗	✓	175	2022-07-29 08:37:23	✓	■ </> ▤ ◉
Event Publish	event	low	This trigger is called just before a MISP Event starts the publishing process	1	✓	✓	180	2022-07-29 12:14:10	✓	■ </> ▤ ◉
Object After Save	object	high ?	This trigger is called after an Object has been saved in the database	35	✗	✓	161	2022-07-28 13:59:37	✗	▶ </> ▤ ◉
Post After Save	post	low	This trigger is called after a Post has been saved in the database	36	✗	✗	176	2022-07-28 13:59:51	✓	■ </> ▤ ◉
User After Save	user	low	This trigger is called after a user has been saved in the database	55	✗	✗	159	2022-07-28 14:00:03	✓	■ </> ▤ ◉
User Before Save	user	low	This trigger is called just before a user is save in the database	42	✓	✗	158	2022-07-28 14:00:32	✓	■ </> ▤ ◉

Workflow - Getting started

Getting started with workflows (1)

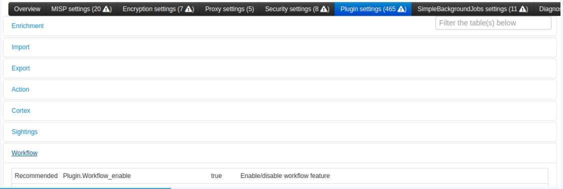
Review MISP settings:

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



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.

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



Recommended	Plugin.Workflow_enable	true	Enable/disable workflow feature

Getting started with workflows (2)

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

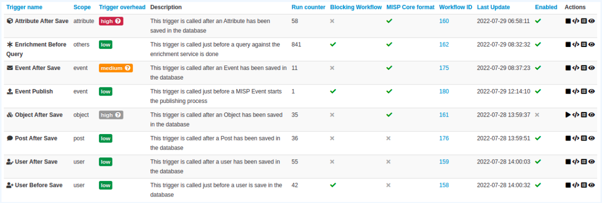
```
# This command should show all `action` modules
$ curl -s http://127.0.0.1:6677/modules | \
jq '.[] | select(.meta."module-type"[] | contains("action")) |
{name: .name, version: .meta.version}'
```

Getting started with workflows (3)

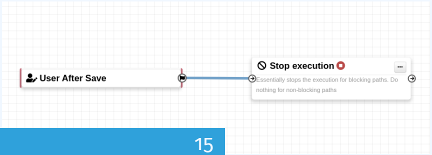
1. Go to the list of modules
   - `Administration > Workflows > List Modules`
   - or `/workflows/moduleIndex`
2. Make sure **default** modules are loaded
3. [optional:misp-module] Make sure **misp-module** modules are loaded

Creating a workflow with the editor

1. Go to the list of triggers `Administration > Workflows`
2. Enable and edit a `trigger` from the list
3. Drag an `action` module from the side panel to the canvas
4. From the `trigger` output, drag an arrow into the `action`'s input (left side)
5. Execute the action that would run the trigger and observe the effect!



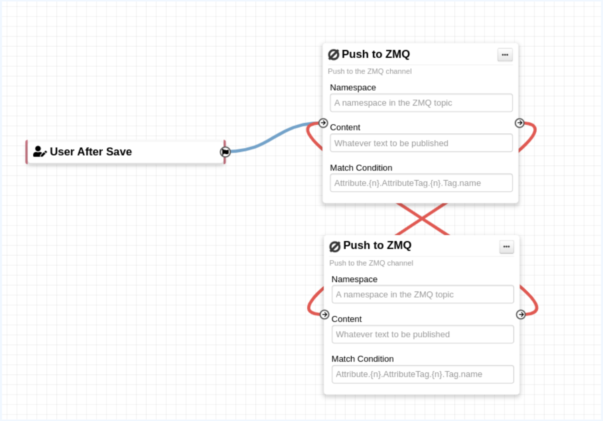
Trigger name	Scope	Trigger overhead	Description	Run counter	Blocking Workflow	MISP Core format	Workflow ID	Last Update	Enabled	Actions
Attribute After Save	attribute	high	This trigger is called after an Attribute has been saved in the database	58	×	✓	160	2022-07-29 06:58:11	✓	
Enrichment Before Query	others	low	This trigger is called just before a query against the enrichment service is done	841	✓	✓	162	2022-07-29 08:32:32	✓	
Event After Save	event	medium	This trigger is called after an Event has been saved in the database	11	×	✓	175	2022-07-29 08:37:23	✓	
Event Publish	event	low	This trigger is called just before a MISP Event starts the publishing process	1	✓	✓	180	2022-07-29 12:14:10	✓	
Object After Save	object	high	This trigger is called after an Object has been saved in the database	35	×	✓	161	2022-07-28 13:59:37	×	
Post After Save	post	low	This trigger is called after a Post has been saved in the database	36	×	×	176	2022-07-28 13:59:51	✓	
User After Save	user	low	This trigger is called after a user has been saved in the database	55	×	×	159	2022-07-28 14:00:03	✓	
User Before Save	user	low	This trigger is called just before a user is save in the database	42	✓	×	158	2022-07-28 14:00:32	✓	


User After Save
Stop execution
Essentially stops the execution for blocking paths. Do nothing for non-blocking paths

Working with the editor

Operations not allowed:

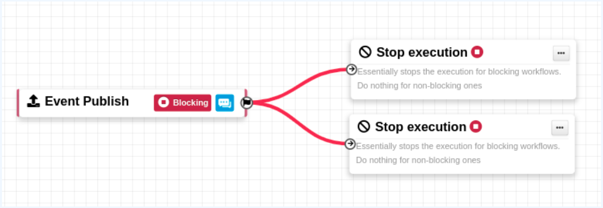
- Execution loop are not authorized
  - Current caveat: If an action re-run the workflow in any way


User After Save
Push to ZMQ
Push to the ZMQ channel
Namespace
A namespace in the ZMQ topic
Content
Whatever text to be published
Match Condition
Attribute.{n}.AttributeTag.{n}.Tag.name
Push to ZMQ
Push to the ZMQ channel
Namespace
A namespace in the ZMQ topic
Content
Whatever text to be published
Match Condition
Attribute.{n}.AttributeTag.{n}.Tag.name

Working with the editor

Operations not allowed:

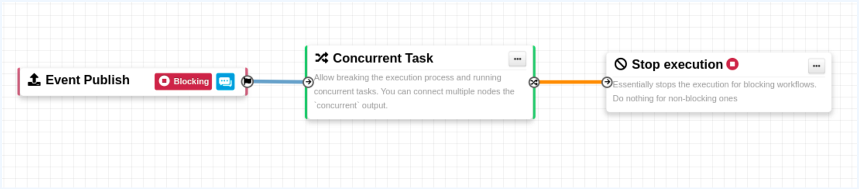
- Multiple connections from the same output
  - Execution order not guaranted and confusing for users


Event Publish
Blocking
Stop execution
Essentially stops the execution for blocking workflows.
Do nothing for non-blocking ones
Stop execution
Essentially stops the execution for blocking workflows.
Do nothing for non-blocking ones

Working with the editor

Operations showing a warning:

- **Blocking** modules after a **concurrent tasks** module

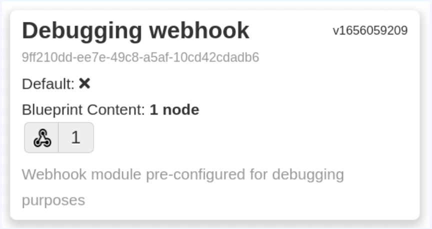


Allow breaking the execution process and running concurrent tasks. You can connect multiple nodes the `concurrent` output.

Essentially stops the execution for blocking workflows. Do nothing for non-blocking ones

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: ✗

Blueprint Content: **1 node**

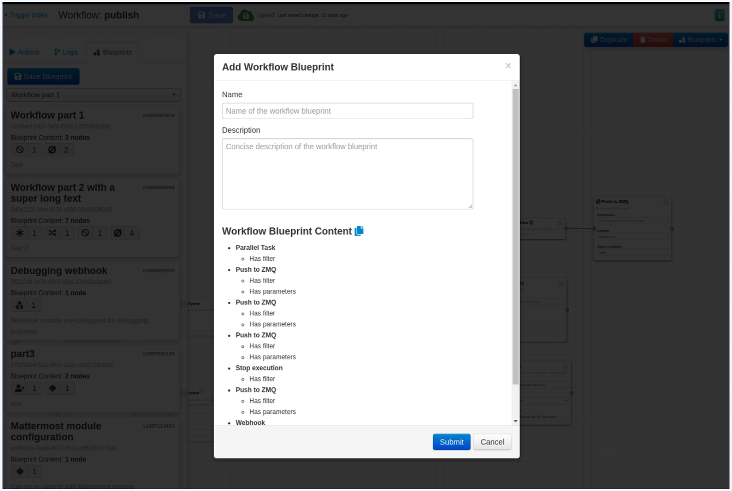
Webhook module pre-configured for debugging purposes

Blueprints origins:

1. From the "official" `misp-workflow-blueprints` repository
2. Created or imported by users

Workflow blueprints: Create

Select one or more modules to be saved as blueprint then click on the `save blueprint` button



Workflow part 1 v1655887874
c1f48aef-041c-4e5f-87b8-525f5453c30a
Blueprint Content: **3 nodes**
1 2
Test

Workflow part 2 with a super long text v1655889258
8d9a1126-1fee-4239-a698-83b00d3f6fd5
Blueprint Content: **7 nodes**
1 1 1 4
Test 2

Debugging webhook v1656059209
9ff210dd-ee7e-49c8-a5af-10cd42cdadb6
Blueprint Content: **1 node**
1
Webhook module pre-configured for debugging purposes

part3 v1657030139
879253b4-f92e-493c-a18c-cdd2133b80fe
Blueprint Content: **2 nodes**
1 1
test

Mattermost module configuration v1657112451
ec9b43ca-9aa9-4673-951b-b982d91770a0
Blueprint Content: **1 node**
1
Can be re-used to add Mattermost logging

Add Workflow Blueprint

Name
Name of the workflow blueprint

Description
Concise description of the workflow blueprint

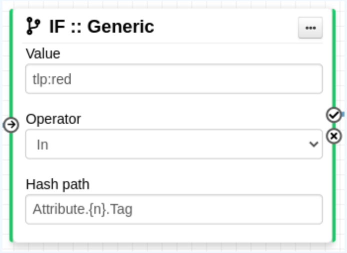
Workflow Blueprint Content

- **Parallel Task**
  - Has filter
- **Push to ZMQ**
  - Has filter
  - Has parameters
- **Push to ZMQ**
  - Has filter
  - Has parameters
- **Push to ZMQ**
  - Has filter
  - Has parameters
- **Stop execution**
  - Has filter
- **Push to ZMQ**
  - Has filter
  - Has parameters
- **Webhook**

Hash path filtering

- Some modules have the possibility to filter or check conditions using CakePHP's path expression.

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



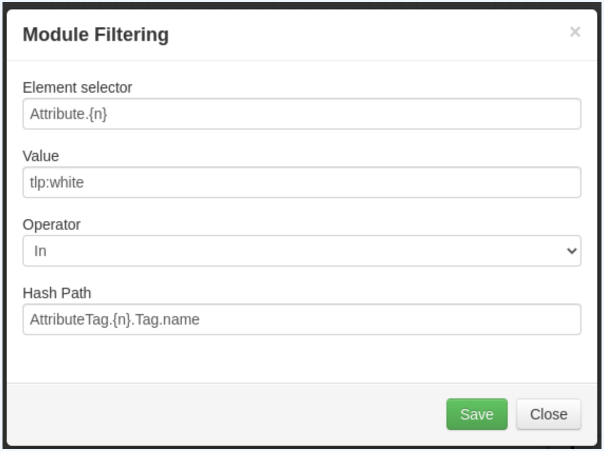
Value: tlp:red

Operator: In

Hash path: Attribute.{n}.Tag

Module filtering

- Some action modules accept **filtering** conditions
- E.g. the `enrich-event` module will only perform the enrichment on Attributes having a `tlp:white` Tag



Module Filtering

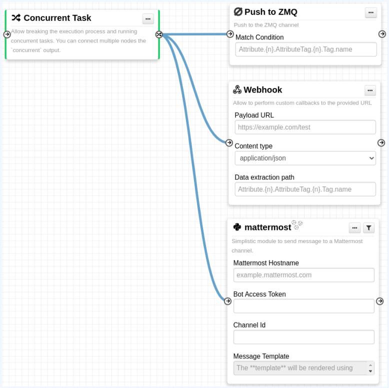
Element selector	Attribute.{n}
Value	tlp:white
Operator	In
Hash Path	AttributeTag.{n}.Tag.name

Data format in Workflows

- All triggers will inject data in a workflow
- In some cases, there is no format (e.g. User after-save)
- In others, the format is **compliant with the MISP Core format**
- In addition to the RFC, the passed 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

- Special type of **logic** module allowing multiple connections
- Allows breaking the execution flow into a **concurrent tasks** to be executed later on by a background worker
- As a side effect, blocking modules **cannot cancel** an ongoing operation anymore



Concurrent Task
Allow breaking the execution process and running concurrent tasks. You can connect multiple nodes the `concurrent` output.

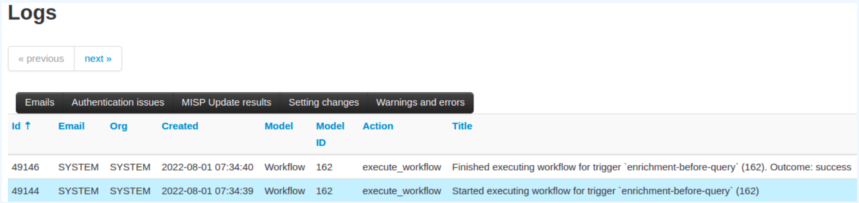
Push to ZMQ
Push to the ZMQ channel
Match Condition: Attribute.{n}.AttributeTag.{n}.Tag.name

Webhook
Allow to perform custom callbacks to the provided URL
Payload URL: https://example.com/test
Content type: application/json
Data extraction path: Attribute.{n}.AttributeTag.{n}.Tag.name

mattermost
Simplistic module to send message to a Mattermost channel.
Mattermost Hostname: example.mattermost.com
Bot Access Token:
Channel Id:
Message Template: The **template** will be rendered using

Debugging Workflows

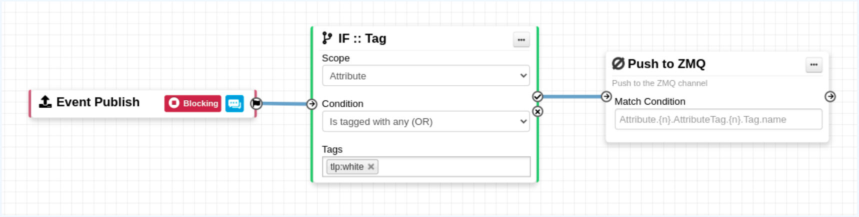
- Workflow execution is logged in the application logs:
  - `/admin/logs/index`
- Or stored on disk in the following file:
  - `/app/tmp/logs/workflow-execution.log`



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)

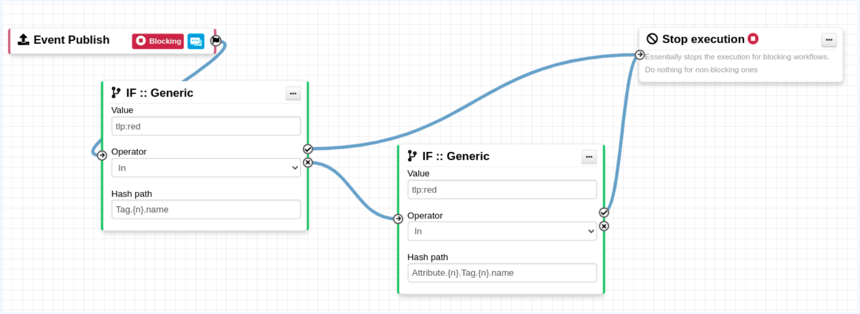
Learning by examples

WORKFLOW EXAMPLE 1



1. The `Event-Publish` trigger uses the MISP core format
2. The `IF::Tag` module checks if at least one of the Attribute has the `tlp:white` tag
3. If it does, the `Push-to-ZMQ` module will be executed

Workflow example 2



Essentially stops the execution for blocking workflows.
Do nothing for non-blocking ones

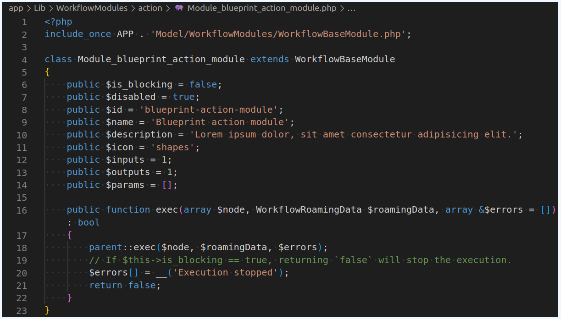
Value: tlp:red
Operator: In
Hash path: Tag.{n}.name

Value: tlp:red
Operator: In
Hash path: Attribute.{n}.Tag.{n}.name

- If an event has the `tlp:red` tag or any of the attribute has it, the publish process will be cancelled

Extending the system

Creating a new module in PHP


app > Lib > WorkflowModules > action > Module_blueprint_action_module.php > ...

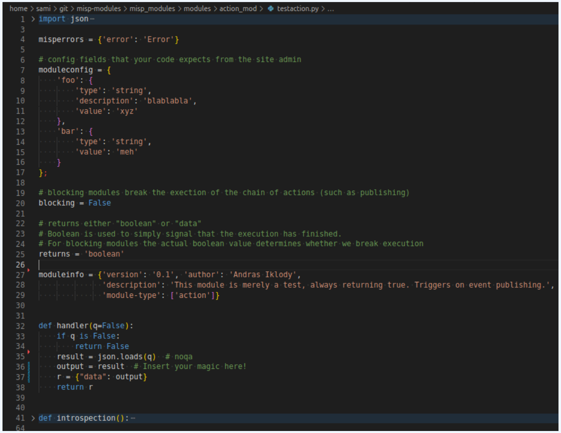
```php
<?php
include_once APP . 'Model/WorkflowModules/WorkflowBaseModule.php';

class Module_blueprint_action_module extends WorkflowBaseModule
{
    public $is_blocking = false;
    public $disabled = true;
    public $id = 'blueprint-action-module';
    public $name = 'Blueprint action module';
    public $description = 'Lorem ipsum dolor, sit amet consectetur adipisicing elit.';
    public $icon = 'shapes';
    public $inputs = 1;
    public $outputs = 1;
    public $params = [];

    public function exec(array $node, WorkflowRoamingData $roamingData, array &$errors = [])
    : bool
    {
        parent::exec($node, $roamingData, $errors);
        // If $this->is_blocking == true, returning `false` will stop the execution.
        $errors[] = __('Execution stopped');
        return false;
    }
}
```

- Module configuration are defined as public variables
- The exec function has to be implemented.
  - If it returns **true**, execution will proceed
  - If it returns **false**
    - And the module is blocking, the execution will stop and the operation will be blocked
    - And the module is not blocking, the execution for the current path will be stopped

Creating a new module in Python


home > sami > git > misp-modules > misp_modules > modules > action_mod > testaction.py > ...

```python
import json

misperrors = {'error': 'Error'}

# config fields that your code expects from the site admin
moduleconfig = {
    'foo': {
        'type': 'string',
        'description': 'blablabla',
        'value': 'xyz'
    },
    'bar': {
        'type': 'string',
        'value': 'meh'
    }
};

# blocking modules break the exection of the chain of actions (such as publishing)
blocking = False

# returns either "boolean" or "data"
# Boolean is used to simply signal that the execution has finished.
# For blocking modules the actual boolean value determines whether we break execution
returns = 'boolean'

moduleinfo = {'version': '0.1', 'author': 'Andras Iklody',
              'description': 'This module is merely a test, always returning true. Triggers on event publishing.',
              'module-type': ['action']}


def handler(q=False):
    if q is False:
        return False
    result = json.loads(q)  # noqa
    output = result  # Insert your magic here!
    r = {"data": output}
    return r


def introspection():
```

- Module configuration are defined in the `moduleinfo` and `moduleconfig` variables
- The `handler` function has to be implemented.
- Blocking logic is the same as other modules