An Introduction to Workflows in MISP

MISP - Threat Sharing

CIRCL / Team MISP Project

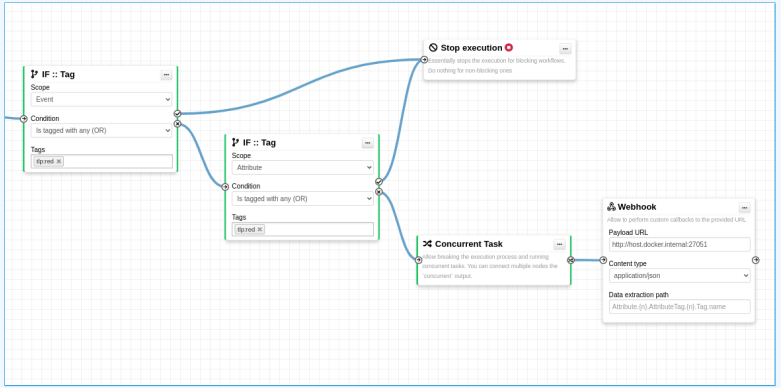
MISP Project
https://www.misp-project.org/

NSPA

An Introduction to Workflows in MISP
MISP - Threat Sharing
CIRCL / Team MISP Project
MISP Project
https://www.misp-project.org/
NSPA
MISP
Threat Sharing

Content of the presentation

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



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

What problems are we trying to tackle

- Initial idea came during GeekWeek7.5[1] 
- Needs:
  - Prevent default MISP behaviors
  - Hook specific actions to run callbacks
- Use-cases:
  - Prevent publication of events not meeting some criterias
  - Prevent querying thrid-party services (e.g. virustotal) with sensitive information
  - Send notifications in a chat rooms
  - And much much more..

[1]Workshop organized by the Canadian Cyber Center

What problems are we trying to tackle
- Initial idea came during GeekWeek7.5[1]
- Needs:
  - Prevent default MISP behaviors
  - Hook specific actions to run callbacks
- Use-cases:
  - Prevent publication of events not meeting some criterias
  - Prevent querying thrid-party services (e.g. virustotal) with sensitive information
  - Send notifications in a chat rooms
  - And much much more..

[1]Workshop organized by the Canadian Cyber Center

Workflow - Fundamentals

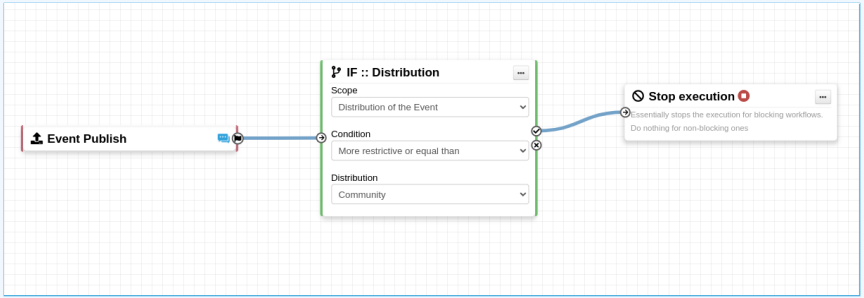
Simplistic overview of a Workflow in action

1. An **action** happens 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

Simplistic overview of a Workflow in action
1. An **action** happens 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



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

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`: Continue 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**: Continue the publishing action
   - **failure** | `blocked`: Stop publishing and log the reason

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: Continue 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: Continue 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 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** can still stop the execution

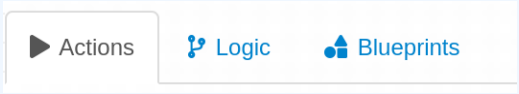
Blocking and non-blocking Workflows
Currently 2 types of workflows:
Blocking: Completion of the action can be prevented
If a blocking module blocks the action
If a blocking module raises an exception
Non-blocking: Workflow execution outcome has no impact
Blocking modules can still stop the execution

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**

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 functions, 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 its module's configuration

Classes of Workflow modules
Actions Logic Blueprints
3 classes of modules
action: Allow to executes functions, 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 its module's configuration

Sources of Workflow modules

3 sources of action modules

- Built-in **default** modules
  - Part of the MISP codebase
  - **app/Model/**WorkflowModules/action/[module_name].php
- User-defined **custom** modules
  - Written in PHP
  - Can extend existing default modules
  - Can use MISP's built-in functionalities (restsearch, enrichment, push to zmq, ...)
  - Faster and easier to implement new complex behaviors
  - **app/Lib/**WorkflowModules/action/[module_name].php

Sources of Workflow modules
3 sources of action modules
Built-in default modules
Part of the MISP codebase
app/Model/WorkflowModules/action/[module_name].php
User-defined custom modules
Written in PHP
Can extend existing default modules
Can use MISP's built-in functionalities (restsearch, enrichment, push to zmq, ...)
Faster and easier to implement new complex behaviors
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* 
  - Written in Python
  - Can use any python libraries
  - New *misp-module* module type: `action`

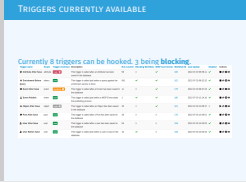
→ Both the PHP and Python systems are **plug-and-play**

Sources of Workflow modules
3 sources of action modules
Modules from the enrichment service
Default and custom modules
From the misp-module
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	✓	

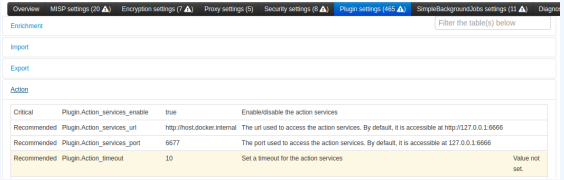


Workflow - Getting started

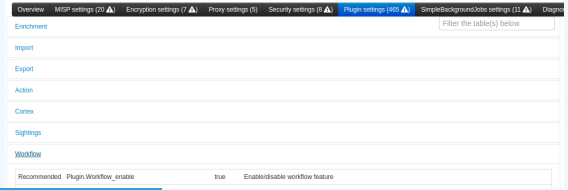
Getting started with workflows (1)

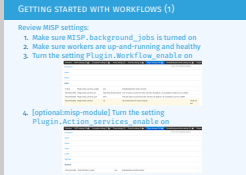
Review MISP settings:

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



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



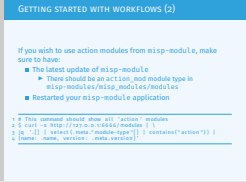


Getting started with workflows (2)

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

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

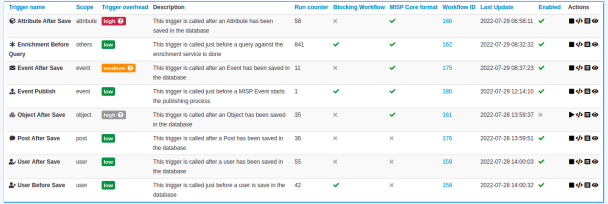
Getting started with workflows (3)

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

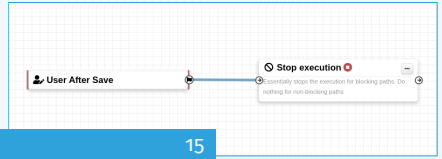
Getting started with workflows (3)
1. Go to the list of modules
   - Administration > Workflows > List Modules
   - or /workflows/moduleIndex
2. Make sure **default** modules are loaded
3. [optional:misp-module] Make sure **misp-module** modules are loaded

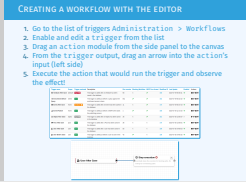
Creating a workflow with the editor

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	✓	

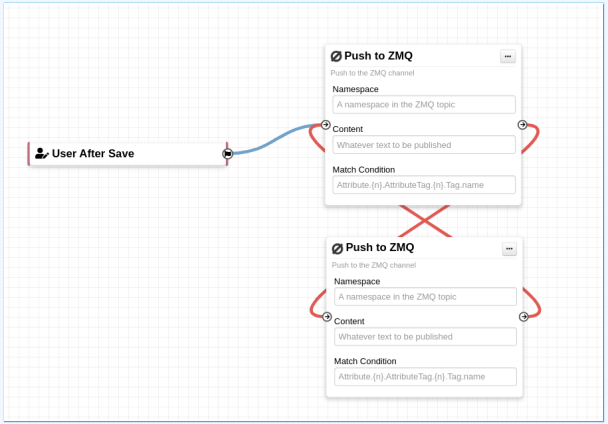


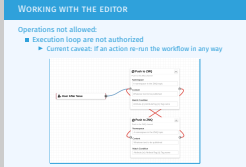


Working with the editor

Operations not allowed:

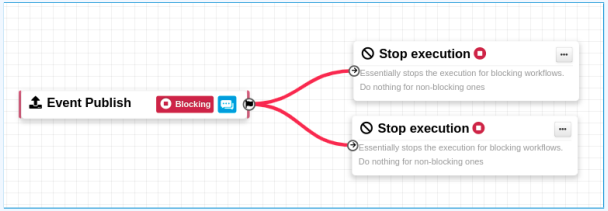
- Execution loop are not authorized
  - Current caveat: If an action re-run the workflow in any way





Operations not allowed:

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

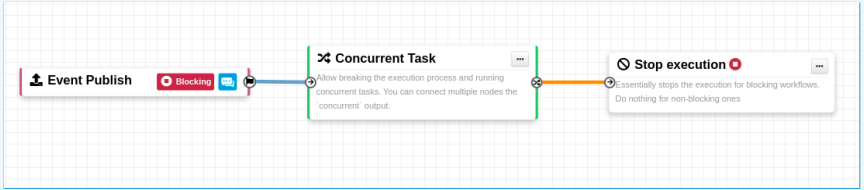


Working with the editor
Operations not allowed:
Multiple connections from the same output
Execution order not guaranted and confusing for users

Working with the editor

Operations showing a warning:

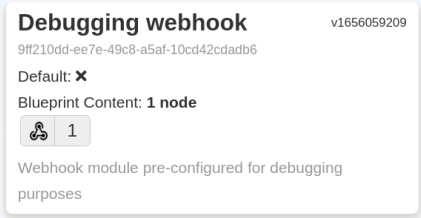
- **Blocking** modules after a **concurrent tasks** module
- **Blocking** modules in a **non-blocking** workflow



Working with the editor
Operations showing a warning:
Blocking modules after a concurrent tasks module
Blocking modules in a non-blocking workflow

Workflow blueprints

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

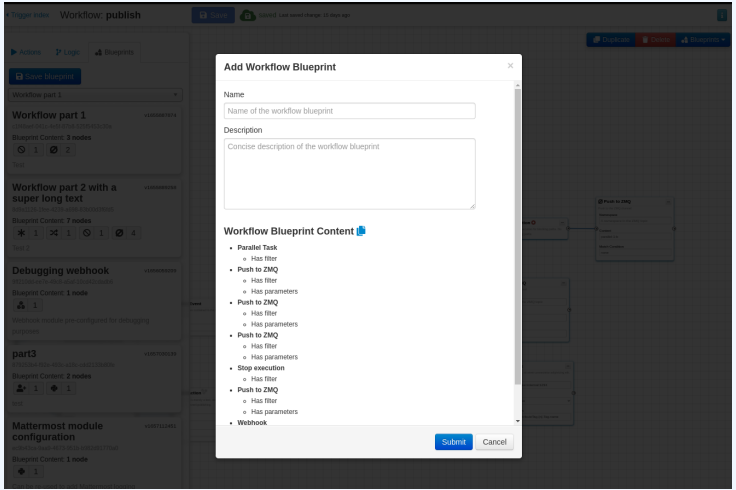


Blueprints origins:

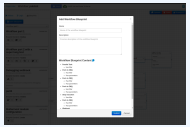
1. From the "official" `misp-workflow-blueprints` repository
2. Created or imported by users

Workflow blueprints: Create

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



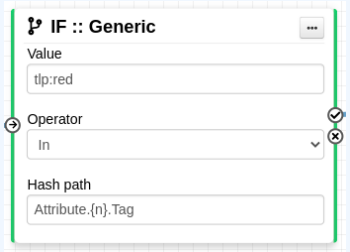
Workflow blueprints: Create
Select one or more modules to be saved as blueprint then click on the save blueprint button

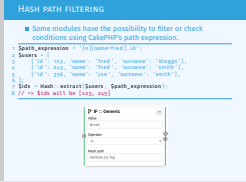


Hash path filtering

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

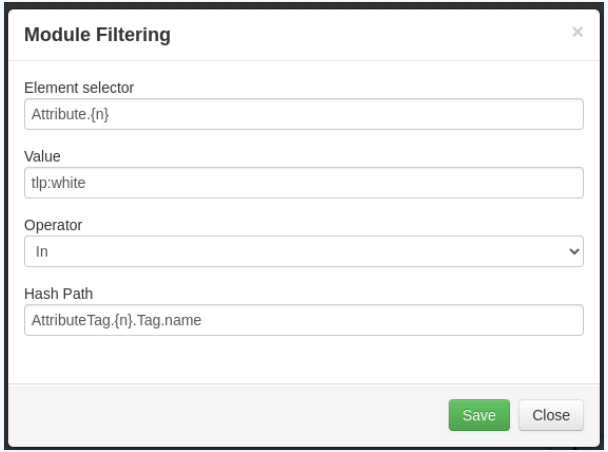
```php
$path_expression = '{n}[name=fred].id';
$users = [
    {'id': 123, 'name': 'fred', 'surname': 'bloggs'},
    {'id': 245, 'name': 'fred', 'surname': 'smith'},
    {'id': 356, 'name': 'joe', 'surname': 'smith'},
];
$ids = Hash::extract($users, $path_expression);
// => $ids will be [123, 245]
```





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

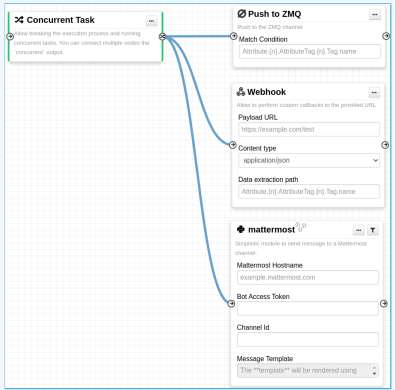


Data format in Workflows

- 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** ongoing operations



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** ongoing operations

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`
- Use the `webhook-listener.py` tool
  - `/app/tools/misp-workflows/webhook-listener.py`

Logs

« previous | next »

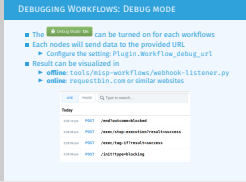
Emails | Authentication issues | MISP Update results | Setting changes | Warnings and errors

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



Learning by examples

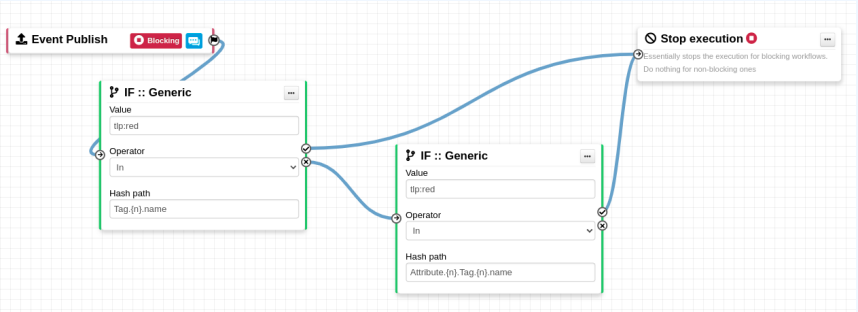
Workflow example 1

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 1

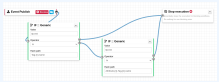
1. The `Event-Publish` trigger uses the MISP core format
2. The `IF::Tag` module checks if at least one of the Attribute has the `tlp:white` tag
3. If it does, the `Push-to-ZMQ` module will be executed

Workflow example 2



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

Workflow example 2



- 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 > ...

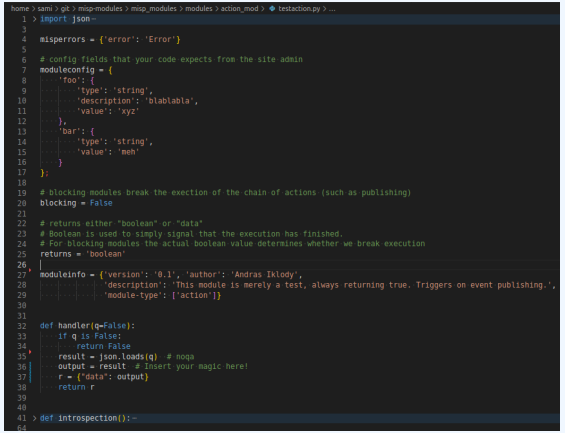
```php
<?php
include_once APP . 'Model/WorkflowModules/WorkflowBaseModule.php';

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;
    }
}
```

- **app/Lib/**`WorkflowModules/action/[module_name].php`
- 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

Creating a new module in PHP

- app/Lib/WorkflowModules/action/[module_name].php
- 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

Creating a new module in Python



- 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

Creating a new module in Python
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