

---

# PyMISP Documentation

*Release 2.4.77*

**Raphaël Vinot**

Nov 22, 2017



---

# Contents

---

<b>1</b>	<b>README</b>	<b>3</b>
<b>2</b>	<b>PyMISP - Python Library to access MISP</b>	<b>5</b>
2.1	Requirements . . . . .	5
2.2	Install from pip . . . . .	5
2.3	Install the latest version from repo . . . . .	5
2.4	Samples and how to use PyMISP . . . . .	5
2.5	Debugging . . . . .	6
2.6	Documentation . . . . .	6
2.7	Everything is a Mutable Mapping . . . . .	6
2.8	MISP Objects . . . . .	7
<b>3</b>	<b>pymisp</b>	<b>9</b>
3.1	pymisp package . . . . .	9
<b>4</b>	<b>Indices and tables</b>	<b>19</b>
	<b>Python Module Index</b>	<b>21</b>



Contents:



# CHAPTER 1

---

README

---





---

## PyMISP - Python Library to access MISP

---

PyMISP is a Python library to access [MISP](#) platforms via their REST API.

PyMISP allows you to fetch events, add or update events/attributes, add or update samples or search for attributes.

### 2.1 Requirements

- requests

### 2.2 Install from pip

```
pip3 install pymisp
```

### 2.3 Install the latest version from repo

```
git clone https://github.com/MISP/PyMISP.git && cd PyMISP  
pip3 install -I .
```

### 2.4 Samples and how to use PyMISP

Various examples and samples scripts are in the [examples/](#) directory.

In the examples directory, you will need to change the `keys.py.sample` to enter your MISP url and API key.

```
cd examples  
cp keys.py.sample keys.py  
vim keys.py
```

The API key of MISP is available in the Automation section of the MISP web interface.

To test if your URL and API keys are correct, you can test with `examples/last.py` to fetch the last 10 events published.

```
cd examples
python3 last.py -l 10
```

## 2.5 Debugging

You have two options there:

1. Pass `debug=True` to PyMISP and it will enable `logging.DEBUG` to `stderr` on the whole module
2. Use the python logging module directly:

```
import logging
logger = logging.getLogger('pymisp')

# Configure it as you wish, for example, enable DEBUG mode:
logger.setLevel(logging.DEBUG)
```

Or if you want to write the debug output to a file instead of `stderr`:

```
import pymisp
import logging

logger = logging.getLogger('pymisp')
logging.basicConfig(level=logging.DEBUG, filename="debug.log", filemode='w',
    ↪format=pymisp.FORMAT)
```

## 2.6 Documentation

PyMISP API documentation is available.

Documentation can be generated with `epydoc`:

```
epydoc --url https://github.com/MISP/PyMISP --graph all --name PyMISP --pdf pymisp -o ↪
    ↪doc
```

## 2.7 Everything is a Mutable Mapping

... or at least everything that can be imported/exported from/to a json blob

`AbstractMISP` is the master class, and inherit `collections.MutableMapping` which means the class can be represented as a python dictionary.

The abstraction assumes every property that should not be seen in the dictionary is prepended with a `_`, or its name is added to the private list `__not_jsonable` (accessible through `update_not_jsonable` and `set_not_jsonable`).

This master class has helpers that will make it easy to load, and export, to, and from, a json string.

MISPEvent, MISPAtribute, MISPObjctReference, MISPObjctAttribute, and MISPObjct are subclasses of AbstractMISP, which mean that they can be handled as python dictionaries.

## 2.8 MISP Objects

Creating a new MISP object generator should be done using a pre-defined template and inherit AbstractMISPObjctGenerator.

Your new MISPObjct generator need to generate attributes, and add them as class properties using add\_attribute.

When the object is sent to MISP, all the class properties will be exported to the JSON export.



## 3.1 pymisp package

### 3.1.1 Submodules

### 3.1.2 pymisp.api module

Python API using the REST interface of MISP

```
class pymisp.api.PyMISP (url, key, ssl=True, out_type='json', debug=None, proxies=None, cert=None,
                        async=False)
```

Bases: `object`

Python API for MISP

#### Parameters

- **url** – URL of the MISP instance you want to connect to
- **key** – API key of the user you want to use
- **ssl** – can be True or False (to check or not the validity

of the certificate. Or a CA\_BUNDLE in case of self signed certificate (the concatenation of all the \*.crt of the chain) :param out\_type: Type of object (json) NOTE: XML output isn't supported anymore, keeping the flag for compatibility reasons. :param debug: Write all the debug information to stderr :param proxies: Proxy dict as describes here: <http://docs.python-requests.org/en/master/user/advanced/#proxies> :param cert: Client certificate, as described there: <http://docs.python-requests.org/en/master/user/advanced/#ssl-cert-verification> :param async: Use asynchronous processing where possible

```
add_attachment (event, attachment, category='Artifacts dropped', to_ids=False, comment=None,
                distribution=None, proposal=False, **kwargs)
```

Add an attachment to the MISP event

#### Parameters

- **event** – The event to add an attachment to

- **attachment** – Either a file handle or a path to a file - will be uploaded

**add\_detection\_name** (*event, name, category='Antivirus detection', to\_ids=False, comment=None, distribution=None, proposal=False, \*\*kwargs*)  
 Add AV detection name(s)

**add\_domain** (*event, domain, category='Network activity', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)  
 Add domain(s)

**add\_domain\_ip** (*event, domain, ip, category='Network activity', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)  
 Add domainlip

**add\_domains\_ips** (*event, domain\_ips, category='Network activity', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)  
 Add multiple domainlip

**add\_email\_attachment** (*event, email, category='Payload delivery', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)  
 Add an email attachment

**add\_email\_dst** (*event, email, category='Payload delivery', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)  
 Add a destination email

**add\_email\_src** (*event, email, category='Payload delivery', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)  
 Add a source email

**add\_email\_subject** (*event, email, category='Payload delivery', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)  
 Add an email subject

**add\_event** (*event*)  
 Add a new event

**Parameters event** – Event as JSON object / string to add

**add\_filename** (*event, filename, category='Artifacts dropped', to\_ids=False, comment=None, distribution=None, proposal=False, \*\*kwargs*)  
 Add filename(s)

**add\_hashes** (*event, category='Artifacts dropped', filename=None, md5=None, sha1=None, sha256=None, ssdeep=None, comment=None, to\_ids=True, distribution=None, proposal=False, \*\*kwargs*)  
 Add hashe(s) to an existing event

**add\_hostname** (*event, hostname, category='Network activity', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)  
 Add hostname(s)

**add\_internal\_comment** (*event, reference, category='Internal reference', to\_ids=False, comment=None, distribution=None, proposal=False, \*\*kwargs*)  
 Add an internal comment

**add\_internal\_link** (*event, reference, category='Internal reference', to\_ids=False, comment=None, distribution=None, proposal=False, \*\*kwargs*)  
 Add an internal link

**add\_internal\_other** (*event, reference, category='Internal reference', to\_ids=False, comment=None, distribution=None, proposal=False, \*\*kwargs*)  
 Add an internal reference (type other)

**add\_internal\_text** (*event, reference, category='Internal reference', to\_ids=False, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add an internal text

**add\_ipdst** (*event, ipdst, category='Network activity', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add destination IP(s)

**add\_ipsrc** (*event, ipsrc, category='Network activity', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add source IP(s)

**add\_mutex** (*event, mutex, category='Artifacts dropped', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add mutex(es)

**add\_named\_attribute** (*event, type\_value, value, category=None, to\_ids=False, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add one or more attributes to an existing event

**add\_net\_other** (*event, netother, category='Network activity', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add a free text entry

**add\_object** (*event\_id, template\_id, misp\_object*)

Add an object

**add\_object\_reference** (*misp\_object\_reference*)

Add a reference to an object

**add\_organisation** (*name, \*\*kwargs*)

**add\_organisation\_json** (*json\_file*)

**add\_pattern** (*event, pattern, in\_file=True, in\_memory=False, category='Artifacts dropped', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add a pattern(s) in file or in memory

**add\_pipe** (*event, named\_pipe, category='Artifacts dropped', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add pipes(s)

**add\_regkey** (*event, regkey, rvalue=None, category='Artifacts dropped', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add a registry key

**add\_regkeys** (*event, regkeys\_values, category='Artifacts dropped', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add a registry keys

**add\_server** (*url, name, authkey, organisation, internal=None, push=False, pull=False, self\_signed=False, push\_rules='', pull\_rules='', submitted\_cert=None, submitted\_client\_cert=None*)

**add\_server\_json** (*json\_file*)

**add\_snort** (*event, snort, category='Network activity', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add SNORT rule(s)

**add\_tag** (*event, tag, attribute=False*)

**add\_target\_email** (*event, target, category='Targeting data', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add an target email

**add\_target\_external** (*event, target, category='Targeting data', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add an target external

**add\_target\_location** (*event, target, category='Targeting data', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add an target location

**add\_target\_machine** (*event, target, category='Targeting data', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add an target machine

**add\_target\_org** (*event, target, category='Targeting data', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add an target organisation

**add\_target\_user** (*event, target, category='Targeting data', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add an target user

**add\_threat\_actor** (*event, target, category='Attribution', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add an threat actor

**add\_traffic\_pattern** (*event, pattern, category='Network activity', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add pattern(s) in traffic

**add\_url** (*event, url, category='Network activity', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add url(s)

**add\_user** (*email, org\_id, role\_id, \*\*kwargs*)

**add\_user\_json** (*json\_file*)

**add\_useragent** (*event, useragent, category='Network activity', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add user agent(s)

**add\_yara** (*event, yara, category='Payload delivery', to\_ids=False, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add yara rule(es)

**av\_detection\_link** (*event, link, category='Antivirus detection', to\_ids=False, comment=None, distribution=None, proposal=False, \*\*kwargs*)

Add AV detection link(s)

**cache\_all\_feeds** ()

Alias for cache\_feeds\_all

**cache\_feed** (*feed\_id*)

Cache a specific feed

**cache\_feeds\_all** ()

Cache all the feeds

**cache\_feeds\_freetext** ()

Cache all the freetext feeds

**cache\_feeds\_misp** ()

Cache all the MISP feeds

**change\_sharing\_group** (*event, sharing\_group\_id*)

Change the sharing group of an event



**change\_threat\_level** (*event, threat\_level\_id*)

Change the threat level of an event

**change\_toids** (*attribute\_uuid, to\_ids*)

Change the toids flag

**compare\_feeds** ()

Generate the comparison matrix for all the MISP feeds

**delete\_attribute** (*attribute\_id, hard\_delete=False*)

Delete an attribute by ID

**delete\_event** (*event\_id*)

Delete an event

**Parameters event\_id** – Event id to delete

**delete\_organisation** (*org\_id*)

**delete\_user** (*user\_id*)

**download\_all\_suricata** ()

Download all suricata rules events.

**download\_last** (*last*)

Download the last updated events.

**Parameters last** – can be defined in days, hours, minutes (for example 5d or 12h or 30m)

**download\_samples** (*sample\_hash=None, event\_id=None, all\_samples=False*)

Download samples, by hash or event ID. If there are multiple samples in one event, use the all\_samples switch

**download\_suricata\_rule\_event** (*event\_id*)

Download one suricata rule event.

**Parameters event\_id** – ID of the event to download (same as get)

**edit\_organisation** (*org\_id, \*\*kwargs*)

**edit\_organisation\_json** (*json\_file, org\_id*)

**edit\_server** (*server\_id, url=None, name=None, authkey=None, organisation=None, internal=None, push=False, pull=False, self\_signed=False, push\_rules='', pull\_rules='', submitted\_cert=None, submitted\_client\_cert=None, delete\_cert=None, delete\_client\_cert=None*)

**edit\_server\_json** (*json\_file, server\_id*)

**edit\_user** (*user\_id, \*\*kwargs*)

**edit\_user\_json** (*json\_file, user\_id*)

**fast\_publish** (*event\_id, alert=False*)

Does the same as the publish method, but just try to publish the event even with one single HTTP GET. The default is to not send a mail as it is assumed this method is called on update.

**fetch\_feed** (*feed\_id*)

Fetch one single feed

**flatten\_error\_messages** (*response*)

Dirty dirty method to normalize the error messages between the API calls. Any response containing the a key 'error' or 'errors' failed at some point, we make one single list out of it.

**freetext** (*event\_id, string, adhereToWarninglists=False, distribution=None*)

Pass a text to the freetext importer

**get** (*eid*)  
Get an event by event ID

**get\_all\_attributes\_txt** (*type\_attr*, *tags=False*, *eventId=False*, *allowNonIDS=False*,  
*date\_from=False*, *date\_to=False*, *last=False*, *enforceWarninglist=False*, *allowNotPublished=False*)  
Get all attributes from a specific type as plain text. Only published and IDS flagged attributes are exported, except if stated otherwise.

**get\_all\_tags** (*quiet=False*)  
Get all the tags used on the instance

**get\_api\_version** ()  
Returns the current version of PyMISP installed on the system

**get\_api\_version\_master** ()  
Get the most recent version of PyMISP from github

**get\_attachment** (*attribute\_id*)  
Get an attachment (not a malware sample) by attribute ID. Returns the attachment as a bytestream, or a dictionary containing the error message.

**Parameters** *attribute\_id* – Attribute ID to fetched

**get\_attributes\_statistics** (*context='type'*, *percentage=None*)  
Get attributes statistics from the MISP instance

**get\_event** (*event\_id*)  
Get an event

**Parameters** *event\_id* – Event id to get

**get\_index** (*filters=None*)  
Return the index.

Warning, there's a limit on the number of results

**get\_object\_template\_id** (*object\_uuid*)  
Gets the template ID corresponding the UUID passed as parameter

**get\_object\_templates\_list** ()  
Returns the list of Object templates available on the MISP instance

**get\_organisation** (*organisation\_id*)

**get\_organisation\_fields\_list** ()

**get\_organisations\_list** (*scope='local'*)

**get\_recommended\_api\_version** ()  
Returns the recommended API version from the server

**get\_roles\_list** ()  
Get the list of existing roles

**get\_sharing\_groups** ()  
Get the existing sharing groups

**get\_stix** (\*\**kwargs*)

**get\_stix\_event** (*event\_id=None*, *with\_attachments=False*, *from\_date=False*, *to\_date=False*,  
*tags=False*)  
Get an event/events in STIX format

**get\_tags\_list** ()  
Get the list of existing tags

**get\_tags\_statistics** (*percentage=None, name\_sort=None*)  
Get tags statistics from the MISP instance

**get\_user** (*user\_id*)

**get\_user\_fields\_list** ()

**get\_users\_list** ()

**get\_version** ()  
Returns the version of the instance.

**get\_version\_master** ()  
Get the most recent version from github

**get\_yara** (*event\_id*)  
Get the yara rules from an event

**new\_event** (*distribution=None, threat\_level\_id=None, analysis=None, info=None, date=None, published=False, orgc\_id=None, org\_id=None, sharing\_group\_id=None*)  
Create and add a new event

**new\_tag** (*name=None, colour='#00ace6', exportable=False*)  
Create a new tag

**proposal\_accept** (*proposal\_id*)  
Accept a proposal

**proposal\_add** (*event\_id, attribute*)  
Add a proposal

**proposal\_discard** (*proposal\_id*)  
Discard a proposal

**proposal\_edit** (*attribute\_id, attribute*)  
Edit a proposal

**proposal\_view** (*event\_id=None, proposal\_id=None*)  
View a proposal

**publish** (*event, alert=True*)  
Publish event (with or without alert email) :param event: pass event or event id (as string or int) to publish  
:param alert: set to True by default (send alerting email) if False will not send alert :return publish status

**pushEventToZMQ** (*event\_id*)  
Force push an event on ZMQ

**remove\_tag** (*event, tag, attribute=False*)

**search** (*controller='events', async\_callback=None, \*\*kwargs*)  
Search via the Rest API

#### Parameters

- **values** – values to search for
- **not\_values** – values *not* to search for
- **type\_attribute** – Type of attribute
- **category** – Category to search
- **org** – Org reporting the event

- **tags** – Tags to search for
- **not\_tags** – Tags *not* to search for
- **date\_from** – First date
- **date\_to** – Last date
- **last** – Last updated events (for example 5d or 12h or 30m)
- **eventid** – Last date
- **withAttachments** – return events with or without the attachments
- **uuid** – search by uuid
- **publish\_timestamp** – the publish timestamp
- **timestamp** – the creation timestamp
- **enforceWarninglist** – Enforce the warning lists
- **searchall** – full text search on the database
- **metadata** – return only metadata if True
- **published** – return only published events
- **to\_ids** – return only the attributes with the to\_ids flag set
- **deleted** – also return the deleted attributes
- **async\_callback** – The function to run when results are returned

**search\_all** (*value*)

Search a value in the whole database

**search\_index** (*published=None, eventid=None, tag=None, datefrom=None, dateuntil=None, eventinfo=None, threatlevel=None, distribution=None, analysis=None, attribute=None, org=None, async\_callback=None, normalize=False*)

Search only at the index level. Use ! in front of value as NOT, default OR. If using async, give a callback that takes 2 args, session and response: basic usage is `pymisp.search_index(..., async_callback=lambda ses,resp: print(resp.json()))`

#### Parameters

- **published** – Published (0,1)
- **eventid** – Event ID(s) | str or list
- **tag** – Tag(s) | str or list
- **datefrom** – First date, in format YYYY-MM-DD
- **dateuntil** – Last date, in format YYYY-MM-DD
- **eventinfo** – Event info(s) to match | str or list
- **threatlevel** – Threat level(s) (1,2,3,4) | str or list
- **distribution** – Distribution level(s) (0,1,2,3) | str or list
- **analysis** – Analysis level(s) (0,1,2) | str or list
- **org** – Organisation(s) | str or list
- **async\_callback** – Function to call when the request returns (if running async)
- **normalize** – Normalize output | True or False

**set\_sightings** (*sightings*)  
Push a sighting (python dictionary)

**sighting\_per\_id** (*attribute\_id*)  
Add a sighting to an attribute (by attribute ID)

**sighting\_per\_json** (*json\_file*)  
Push a sighting (JSON file)

**sighting\_per\_uuid** (*attribute\_uuid*)  
Add a sighting to an attribute (by attribute UUID)

**tag** (*uuid, tag*)  
Tag an event or an attribute

**test\_connection** ()  
Test the auth key

**untag** (*uuid, tag*)  
Untag an event or an attribute

**update** (*event*)  
Update an event by ID

**update\_event** (*event\_id, event*)  
Update an event

#### Parameters

- **event\_id** – Event id to update
- **event** – Event as JSON object / string to add

**upload\_sample** (*filename, filepath\_or\_bytes, event\_id, distribution=None, to\_ids=True, category=None, comment=None, info=None, analysis=None, threat\_level\_id=None*)  
Upload a sample

**upload\_samplelist** (*filepaths, event\_id, distribution=None, to\_ids=True, category=None, comment=None, info=None, analysis=None, threat\_level\_id=None*)  
Upload a list of samples

**view\_feed** (*feed\_ids*)  
Get the content of a single feed

**view\_feeds** ()  
Get the content of all the feeds

`pymisp.api.deprecated` (*func*)

This is a decorator which can be used to mark functions as deprecated. It will result in a warning being emitted when the function is used.

### 3.1.3 Module contents



## CHAPTER 4

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`





**p**

`pymisp`, 17  
`pymisp.api`, 9



## A

add\_attachment() (pymisp.api.PyMISP method), 9  
 add\_detection\_name() (pymisp.api.PyMISP method), 10  
 add\_domain() (pymisp.api.PyMISP method), 10  
 add\_domain\_ip() (pymisp.api.PyMISP method), 10  
 add\_domains\_ips() (pymisp.api.PyMISP method), 10  
 add\_email\_attachment() (pymisp.api.PyMISP method), 10  
 add\_email\_dst() (pymisp.api.PyMISP method), 10  
 add\_email\_src() (pymisp.api.PyMISP method), 10  
 add\_email\_subject() (pymisp.api.PyMISP method), 10  
 add\_event() (pymisp.api.PyMISP method), 10  
 add\_filename() (pymisp.api.PyMISP method), 10  
 add\_hashes() (pymisp.api.PyMISP method), 10  
 add\_hostname() (pymisp.api.PyMISP method), 10  
 add\_internal\_comment() (pymisp.api.PyMISP method), 10  
 add\_internal\_link() (pymisp.api.PyMISP method), 10  
 add\_internal\_other() (pymisp.api.PyMISP method), 10  
 add\_internal\_text() (pymisp.api.PyMISP method), 10  
 add\_ipdst() (pymisp.api.PyMISP method), 11  
 add\_ipsrc() (pymisp.api.PyMISP method), 11  
 add\_mutex() (pymisp.api.PyMISP method), 11  
 add\_named\_attribute() (pymisp.api.PyMISP method), 11  
 add\_net\_other() (pymisp.api.PyMISP method), 11  
 add\_object() (pymisp.api.PyMISP method), 11  
 add\_object\_reference() (pymisp.api.PyMISP method), 11  
 add\_organisation() (pymisp.api.PyMISP method), 11  
 add\_organisation\_json() (pymisp.api.PyMISP method), 11  
 add\_pattern() (pymisp.api.PyMISP method), 11  
 add\_pipe() (pymisp.api.PyMISP method), 11  
 add\_regkey() (pymisp.api.PyMISP method), 11  
 add\_regkeys() (pymisp.api.PyMISP method), 11  
 add\_server() (pymisp.api.PyMISP method), 11  
 add\_server\_json() (pymisp.api.PyMISP method), 11  
 add\_snort() (pymisp.api.PyMISP method), 11  
 add\_tag() (pymisp.api.PyMISP method), 11  
 add\_target\_email() (pymisp.api.PyMISP method), 11

add\_target\_external() (pymisp.api.PyMISP method), 11  
 add\_target\_location() (pymisp.api.PyMISP method), 12  
 add\_target\_machine() (pymisp.api.PyMISP method), 12  
 add\_target\_org() (pymisp.api.PyMISP method), 12  
 add\_target\_user() (pymisp.api.PyMISP method), 12  
 add\_threat\_actor() (pymisp.api.PyMISP method), 12  
 add\_traffic\_pattern() (pymisp.api.PyMISP method), 12  
 add\_url() (pymisp.api.PyMISP method), 12  
 add\_user() (pymisp.api.PyMISP method), 12  
 add\_user\_json() (pymisp.api.PyMISP method), 12  
 add\_useragent() (pymisp.api.PyMISP method), 12  
 add\_yara() (pymisp.api.PyMISP method), 12  
 av\_detection\_link() (pymisp.api.PyMISP method), 12

## C

cache\_all\_feeds() (pymisp.api.PyMISP method), 12  
 cache\_feed() (pymisp.api.PyMISP method), 12  
 cache\_feeds\_all() (pymisp.api.PyMISP method), 12  
 cache\_feeds\_freetext() (pymisp.api.PyMISP method), 12  
 cache\_feeds\_misp() (pymisp.api.PyMISP method), 12  
 change\_sharing\_group() (pymisp.api.PyMISP method), 12  
 change\_threat\_level() (pymisp.api.PyMISP method), 12  
 change\_toids() (pymisp.api.PyMISP method), 13  
 compare\_feeds() (pymisp.api.PyMISP method), 13

## D

delete\_attribute() (pymisp.api.PyMISP method), 13  
 delete\_event() (pymisp.api.PyMISP method), 13  
 delete\_organisation() (pymisp.api.PyMISP method), 13  
 delete\_user() (pymisp.api.PyMISP method), 13  
 deprecated() (in module pymisp.api), 17  
 download\_all\_suricata() (pymisp.api.PyMISP method), 13  
 download\_last() (pymisp.api.PyMISP method), 13  
 download\_samples() (pymisp.api.PyMISP method), 13  
 download\_suricata\_rule\_event() (pymisp.api.PyMISP method), 13

## E

`edit_organisation()` (pymisp.api.PyMISP method), 13  
`edit_organisation_json()` (pymisp.api.PyMISP method), 13  
`edit_server()` (pymisp.api.PyMISP method), 13  
`edit_server_json()` (pymisp.api.PyMISP method), 13  
`edit_user()` (pymisp.api.PyMISP method), 13  
`edit_user_json()` (pymisp.api.PyMISP method), 13

## F

`fast_publish()` (pymisp.api.PyMISP method), 13  
`fetch_feed()` (pymisp.api.PyMISP method), 13  
`flatten_error_messages()` (pymisp.api.PyMISP method), 13  
`freetext()` (pymisp.api.PyMISP method), 13

## G

`get()` (pymisp.api.PyMISP method), 14  
`get_all_attributes_txt()` (pymisp.api.PyMISP method), 14  
`get_all_tags()` (pymisp.api.PyMISP method), 14  
`get_api_version()` (pymisp.api.PyMISP method), 14  
`get_api_version_master()` (pymisp.api.PyMISP method), 14  
`get_attachment()` (pymisp.api.PyMISP method), 14  
`get_attributes_statistics()` (pymisp.api.PyMISP method), 14  
`get_event()` (pymisp.api.PyMISP method), 14  
`get_index()` (pymisp.api.PyMISP method), 14  
`get_object_template_id()` (pymisp.api.PyMISP method), 14  
`get_object_templates_list()` (pymisp.api.PyMISP method), 14  
`get_organisation()` (pymisp.api.PyMISP method), 14  
`get_organisation_fields_list()` (pymisp.api.PyMISP method), 14  
`get_organisations_list()` (pymisp.api.PyMISP method), 14  
`get_recommended_api_version()` (pymisp.api.PyMISP method), 14  
`get_roles_list()` (pymisp.api.PyMISP method), 14  
`get_sharing_groups()` (pymisp.api.PyMISP method), 14  
`get_stix()` (pymisp.api.PyMISP method), 14  
`get_stix_event()` (pymisp.api.PyMISP method), 14  
`get_tags_list()` (pymisp.api.PyMISP method), 14  
`get_tags_statistics()` (pymisp.api.PyMISP method), 15  
`get_user()` (pymisp.api.PyMISP method), 15  
`get_user_fields_list()` (pymisp.api.PyMISP method), 15  
`get_users_list()` (pymisp.api.PyMISP method), 15  
`get_version()` (pymisp.api.PyMISP method), 15  
`get_version_master()` (pymisp.api.PyMISP method), 15  
`get_yara()` (pymisp.api.PyMISP method), 15

## N

`new_event()` (pymisp.api.PyMISP method), 15

`new_tag()` (pymisp.api.PyMISP method), 15

## P

`proposal_accept()` (pymisp.api.PyMISP method), 15  
`proposal_add()` (pymisp.api.PyMISP method), 15  
`proposal_discard()` (pymisp.api.PyMISP method), 15  
`proposal_edit()` (pymisp.api.PyMISP method), 15  
`proposal_view()` (pymisp.api.PyMISP method), 15  
`publish()` (pymisp.api.PyMISP method), 15  
`pushEventToZMQ()` (pymisp.api.PyMISP method), 15  
PyMISP (class in pymisp.api), 9  
pymisp (module), 17  
pymisp.api (module), 9

## R

`remove_tag()` (pymisp.api.PyMISP method), 15

## S

`search()` (pymisp.api.PyMISP method), 15  
`search_all()` (pymisp.api.PyMISP method), 16  
`search_index()` (pymisp.api.PyMISP method), 16  
`set_sightings()` (pymisp.api.PyMISP method), 16  
`sighting_per_id()` (pymisp.api.PyMISP method), 17  
`sighting_per_json()` (pymisp.api.PyMISP method), 17  
`sighting_per_uuid()` (pymisp.api.PyMISP method), 17

## T

`tag()` (pymisp.api.PyMISP method), 17  
`test_connection()` (pymisp.api.PyMISP method), 17

## U

`untag()` (pymisp.api.PyMISP method), 17  
`update()` (pymisp.api.PyMISP method), 17  
`update_event()` (pymisp.api.PyMISP method), 17  
`upload_sample()` (pymisp.api.PyMISP method), 17  
`upload_samplelist()` (pymisp.api.PyMISP method), 17

## V

`view_feed()` (pymisp.api.PyMISP method), 17  
`view_feeds()` (pymisp.api.PyMISP method), 17