PyMISP Documentation

Release 2.4.77

Raphaël Vinot

Nov 19, 2017

Contents

1 README **3**

2 PyMISP - Python Library to access MISP **5**
- 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

3 pymisp **9**
- 3.1 pymisp package . . . 9

4 Indices and tables **19**

Python Module Index **21**

Contents:

CHAPTER 1

README

CHAPTER 2

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 whish, 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`, `MISPAttribute`, `MISPObjectReference`, `MISPObjectAttribute`, and `MISPObject` 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 `AbstractMISPObjectGenerator`.

Your new MISPObject 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.

CHAPTER 3

pymisp

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*, *asynch=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 ot not the validity

of the certificate. Or a CA_BUNDLE in case of self signed certiifcate (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 asynch: 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 atachment

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 attachement (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 corresponting 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 ! infront 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** – Evend 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

Python Module Index

p

pymisp, 17
pymisp.api, 9

Index

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