

---

# **PyMISP Documentation**

*Release master*

**Raphaël Vinot**

**Jan 20, 2018**



---

## 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 . . . . .	9
3.2	MISPAbstract . . . . .	17
3.3	MISPEncode . . . . .	18
3.4	MISPEvent . . . . .	18
3.5	MISPAttribute . . . . .	20
3.6	MISPObject . . . . .	21
3.7	MISPObjectAttribute . . . . .	22
3.8	MISPObjectReference . . . . .	23
3.9	MISPTag . . . . .	23
3.10	MISPUser . . . . .	24
3.11	MISPOrganisation . . . . .	25
<b>4</b>	<b>pymisp - Tools</b>	<b>27</b>
4.1	File Object . . . . .	27
4.2	ELF Object . . . . .	28
4.3	PE Object . . . . .	30
4.4	Mach-O Object . . . . .	32
4.5	VT Report Object . . . . .	33
4.6	STIX . . . . .	35
4.7	OpenIOC . . . . .	35
<b>5</b>	<b>Indices and tables</b>	<b>37</b>
	<b>Python Module Index</b>	<b>39</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
git submodule update --init
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`, `MISPAttribute`, `MISPObjReference`, `MISPObjAttribute`, and `MISPObj` 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 `AbstractMISPObjGenerator`.

Your new `MISPObj` 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.



`pymisp.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 PyMISP

**class** `pymisp.PyMISP` (*url, key, ssl=True, out\_type='json', debug=None, proxies=None, cert=None, async=False*)

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

**add\_attachment** (*event, attachment, category='Artifacts dropped', to\_ids=False, comment=None, distribution=None, proposal=False, filename=None, \*\*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
- **filename** – Explicitly defined attachment filename

**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\_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\_snort** (*event, snort, category='Network activity', to\_ids=True, comment=None, distribution=None, proposal=False, \*\*kwargs*)  
Add SNORT rule(s)
- 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\_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\_comment** (*attribute\_uuid, comment*)

Change the comment of attribute

**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\_object** (*id*)

Deletes an object

**delete\_object\_reference** (*id*)

Deletes a reference to an object

**download\_all\_suricata** ()

Download all suricata rules events.

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

Download the last published 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)

**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\_csv** (*eventid=None, attributes=[], object\_attributes=[], misp\_types=[], context=False, ignore=False, last=None*)

Get MISP values in CSV format :param eventid: The event ID to query :param attributes: The column names to export from normal attributes (i.e. uuid, value, type, ...) :param object\_attributes: The column names to export from attributes within objects (i.e. uuid, value, type, ...) :param misp\_types: MISP types to get (i.e. ip-src, hostname, ...) :param context: Add event level context (event\_info,event\_member\_org,event\_source\_org,event\_distribution,event\_threat\_level\_id,event\_analysis,event\_date,event\_date\_published) :param ignore: Returns the attributes even if the event isn't published, or the attribute doesn't have the to\_ids flag set

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

Get an event

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

**get\_events\_last\_modified** (*search\_from, search\_to=None*)

Download the last modified events.

**Parameters**

- **search\_from** – Beginning of the interval. Can be either a timestamp, or a date (2000-12-21)
- **search\_to** – End of the interval. Can be either a timestamp, or a date (2000-12-21)

**get\_index** (*filters=None*)

Return the index.

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

**get\_live\_query\_acl** ()

This should return an empty list, unless the ACL is outdated.

**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\_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\_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\_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, hide\_tag=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

**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 published 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 timestamp of the last modification. Can be a list (from->to)
- **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 or MISPSighting) or a list of sightings

**sighting** (*value, source=None, type=None, timestamp=None, \*\*kwargs*)

Set a single sighting. :value: Value can either be the attribute's value (to update sighting on all the attributes with this value),

or an UUID in order to update the sightings of one particular attribute.

**Source** Source of the sighting

**Type** Type of the sighting

**Timestamp** Timestamp associated to the sighting

**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

## 3.2 MISPAbstract

```
class pymisp.AbstractMISP (**kwargs)
```

**edited**

Recursively check if an object has been edited and update the flag accordingly to the parent objects

**from\_dict** (\*\*kwargs)

Loading all the parameters as class properties, if they aren't *None*. This method aims to be called when all the properties requiring a special treatment are processed. Note: This method is used when you initialize an object with existing data so by default, the class is flagged as not edited.

**from\_json** (*json\_string*)

Load a JSON string

**jsonable** ()

This method is used by the JSON encoder

**properties**

All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the properties starting with a `_` (private), or listed in `__not_jsonable` will be skipped.

**set\_not\_jsonable** (*\*args*)

Set `__not_jsonable` to a new list

**to\_dict** ()

Dump the lass to a dictionary. This method automatically removes the timestamp recursively in every object that has been edited in order to let MISP update the event accordingly.

**to\_json** ()

Dump recursively any class of type `MISPAbstract` to a json string

**update\_not\_jsonable** (*\*args*)

Add entries to the `__not_jsonable` list

### 3.3 MISPEncode

```
class pymisp.MISPEncode (skipkeys=False, ensure_ascii=True, check_circular=True, al-
                        low_nan=True, sort_keys=False, indent=None, separators=None,
                        default=None)
```

### 3.4 MISPEvent

```
class pymisp.MISPEvent (describe_types=None, strict_validation=False)
```

**add\_attribute** (*type, value, \*\*kwargs*)

Add an attribute. *type* and *value* are required but you can pass all other parameters supported by `MISPAtribute`

**add\_attribute\_tag** (*tag, attribute\_identifier*)

Add a tag to an existing attribute, raise an Exception if the attribute doesn't exist. *:tag:* Tag name as a string, `MISPTag` instance, or dictionary *:attribute\_identifier:* can be an ID, UUID, or the value.

**add\_object** (*obj=None, \*\*kwargs*)

Add an object to the Event, either by passing a `MISPObject`, or a dictionary

**add\_proposal** (*shadow\_attribute=None, \*\*kwargs*)

Alias for `add_shadow_attribute`

**add\_shadow\_attribute** (*shadow\_attribute=None, \*\*kwargs*)

Add a tag to the attribute (by name or a `MISPTag` object)

**clear** () → None. Remove all items from D.

**delete\_attribute** (*attribute\_id*)

Delete an attribute, you can search by ID or UUID

**edited**

Recursively check if an object has been edited and update the flag accordingly to the parent objects

**from\_json** (*json\_string*)

Load a JSON string

**get** (*k*, *d*) → *D*[*k*] if *k* in *D*, else *d*. *d* defaults to None.

**get\_attribute\_tag** (*attribute\_identifier*)

Return the tags associated to an attribute or an object attribute. *:attribute\_identifier:* can be an ID, UUID, or the value.

**get\_object\_by\_id** (*object\_id*)

Get an object by ID (the ID is the one set by the server when creating the new object)

**items** () → a set-like object providing a view on *D*'s items

**jsonable** ()

This method is used by the JSON encoder

**keys** () → a set-like object providing a view on *D*'s keys

**load** (*json\_event*)

Load a JSON dump from a pseudo file or a JSON string

**load\_file** (*event\_path*)

Load a JSON dump from a file on the disk

**pop** (*k*, *d*) → *v*, remove specified key and return the corresponding value.

If key is not found, *d* is returned if given, otherwise *KeyError* is raised.

**popitem** () → (*k*, *v*), remove and return some (key, value) pair

as a 2-tuple; but raise *KeyError* if *D* is empty.

**properties**

All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the properties starting with a *\_* (private), or listed in *\_\_not\_jsonable* will be skipped.

**publish** ()

Mark the attribute as published

**set\_date** (*date*, *ignore\_invalid=False*)

Set a date for the event (string, datetime, or date object)

**set\_not\_jsonable** (*\*args*)

Set *\_\_not\_jsonable* to a new list

**setdefault** (*k*, *d*) → *D*.*get*(*k*,*d*), also set *D*[*k*]=*d* if *k* not in *D*

**to\_json** ()

Dump recursively any class of type *MISPAbstract* to a json string

**unpublish** ()

Mark the attribute as un-published (set publish flag to false)

**update** (*[E]*, *\*\*F*) → None. Update *D* from mapping/iterable *E* and *F*.

If *E* present and has a *.keys()* method, does: for *k* in *E*: *D*[*k*] = *E*[*k*] If *E* present and lacks *.keys()* method, does: for (*k*, *v*) in *E*: *D*[*k*] = *v* In either case, this is followed by: for *k*, *v* in *F*.*items*(): *D*[*k*] = *v*

**update\_not\_jsonable** (*\*args*)

Add entries to the *\_\_not\_jsonable* list

**values** () → an object providing a view on *D*'s values

## 3.5 MISPAtribute

**class** pymisp.MISPAtribute (*describe\_types=None, strict=False*)

**add\_proposal** (*shadow\_attribute=None, \*\*kwargs*)

Alias for add\_shadow\_attribute

**add\_shadow\_attribute** (*shadow\_attribute=None, \*\*kwargs*)

Add a tag to the attribute (by name or a MISPTag object)

**clear** () → None. Remove all items from D.

**delete** ()

Mark the attribute as deleted (soft delete)

**edited**

Recursively check if an object has been edited and update the flag accordingly to the parent objects

**from\_json** (*json\_string*)

Load a JSON string

**get** (*k[, d]*) → D[k] if k in D, else d. d defaults to None.

**items** () → a set-like object providing a view on D's items

**jsonable** ()

This method is used by the JSON encoder

**keys** () → a set-like object providing a view on D's keys

**known\_types**

Returns a list of all the known MISP attributes types

**malware\_binary**

Returns a BytesIO of the malware (if the attribute has one, obvs).

**pop** (*k[, d]*) → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise KeyError is raised.

**popitem** () → (k, v), remove and return some (key, value) pair

as a 2-tuple; but raise KeyError if D is empty.

**properties**

All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the properties starting with a \_ (private), or listed in \_\_not\_jsonable will be skipped.

**set\_not\_jsonable** (*\*args*)

Set \_\_not\_jsonable to a new list

**setdefault** (*k[, d]*) → D.get(k,d), also set D[k]=d if k not in D

**to\_json** ()

Dump recursively any class of type MISPAbstract to a json string

**update** (*[E], \*\*F*) → None. Update D from mapping/iterable E and F.

If E present and has a .keys() method, does: for k in E: D[k] = E[k] If E present and lacks .keys() method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v

**update\_not\_jsonable** (*\*args*)

Add entries to the \_\_not\_jsonable list

**values** () → an object providing a view on D's values



## 3.6 MISPObject

**class** pymisp.MISPObject (*name*, *strict=False*, *standalone=False*, *default\_attributes\_parameters={}*, *\*\*kwargs*)

**add\_attribute** (*object\_relation*, *\*\*value*)

Add an attribute. *object\_relation* is required and the value key is a dictionary with all the keys supported by MISPAtribute

**add\_reference** (*referenced\_uuid*, *relationship\_type*, *comment=None*, *\*\*kwargs*)

Add a link (uuid) to an other object

**clear** () → None. Remove all items from D.

**edited**

Recursively check if an object has been edited and update the flag accordingly to the parent objects

**from\_json** (*json\_string*)

Load a JSON string

**get** (*k*, [*d*]) → D[k] if k in D, else d. d defaults to None.

**get\_attributes\_by\_relation** (*object\_relation*)

Returns the list of attributes with the given object relation in the object

**has\_attributes\_by\_relation** (*list\_of\_relations*)

True if all the relations in the list are defined in the object

**items** () → a set-like object providing a view on D's items

**jsonable** ()

This method is used by the JSON encoder

**keys** () → a set-like object providing a view on D's keys

**pop** (*k*, [*d*]) → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise KeyError is raised.

**popitem** () → (k, v), remove and return some (key, value) pair

as a 2-tuple; but raise KeyError if D is empty.

**properties**

All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the properties starting with a \_ (private), or listed in `__not_jsonable` will be skipped.

**set\_not\_jsonable** (*\*args*)

Set `__not_jsonable` to a new list

**setdefault** (*k*, [*d*]) → D.get(k,d), also set D[k]=d if k not in D

**update** ([*E*], *\*\*F*) → None. Update D from mapping/iterable E and F.

If E present and has a `.keys()` method, does: for k in E: D[k] = E[k] If E present and lacks `.keys()` method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v

**update\_not\_jsonable** (*\*args*)

Add entries to the `__not_jsonable` list

**values** () → an object providing a view on D's values

## 3.7 MISPObjectAttribute

**class** pymisp.MISPObjectAttribute (*definition*)

**add\_proposal** (*shadow\_attribute=None, \*\*kwargs*)

Alias for add\_shadow\_attribute

**add\_shadow\_attribute** (*shadow\_attribute=None, \*\*kwargs*)

Add a tag to the attribute (by name or a MISPTag object)

**clear** () → None. Remove all items from D.

**delete** ()

Mark the attribute as deleted (soft delete)

**edited**

Recursively check if an object has been edited and update the flag accordingly to the parent objects

**from\_json** (*json\_string*)

Load a JSON string

**get** (*k[, d]*) → D[k] if k in D, else d. d defaults to None.

**items** () → a set-like object providing a view on D's items

**jsonable** ()

This method is used by the JSON encoder

**keys** () → a set-like object providing a view on D's keys

**known\_types**

Returns a list of all the known MISP attributes types

**malware\_binary**

Returns a BytesIO of the malware (if the attribute has one, obvs).

**pop** (*k[, d]*) → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise KeyError is raised.

**popitem** () → (k, v), remove and return some (key, value) pair

as a 2-tuple; but raise KeyError if D is empty.

**properties**

All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the properties starting with a \_ (private), or listed in \_\_not\_jsonable will be skipped.

**set\_not\_jsonable** (*\*args*)

Set \_\_not\_jsonable to a new list

**setdefault** (*k[, d]*) → D.get(k,d), also set D[k]=d if k not in D

**to\_json** ()

Dump recursively any class of type MISPAbstract to a json string

**update** (*[E], \*\*F*) → None. Update D from mapping/iterable E and F.

If E present and has a .keys() method, does: for k in E: D[k] = E[k] If E present and lacks .keys() method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v

**update\_not\_jsonable** (*\*args*)

Add entries to the \_\_not\_jsonable list

**values** () → an object providing a view on D's values

## 3.8 MISPObjReference

**class** pymisp.MISPObjReference

**clear** () → None. Remove all items from D.

**edited**

Recursively check if an object has been edited and update the flag accordingly to the parent objects

**from\_json** (*json\_string*)

Load a JSON string

**get** (*k*, *d*) → D[k] if k in D, else d. d defaults to None.

**items** () → a set-like object providing a view on D's items

**jsonable** ()

This method is used by the JSON encoder

**keys** () → a set-like object providing a view on D's keys

**pop** (*k*, *d*) → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise KeyError is raised.

**popitem** () → (k, v), remove and return some (key, value) pair

as a 2-tuple; but raise KeyError if D is empty.

**properties**

All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the properties starting with a \_ (private), or listed in \_\_not\_jsonable will be skipped.

**set\_not\_jsonable** (*\*args*)

Set \_\_not\_jsonable to a new list

**setdefault** (*k*, *d*) → D.get(k,d), also set D[k]=d if k not in D

**to\_dict** ()

Dump the lass to a dictionary. This method automatically removes the timestamp recursively in every object that has been edited is order to let MISP update the event accordingly.

**to\_json** ()

Dump recursively any class of type MISPAbstract to a json string

**update** (*[E]*, *\*\*F*) → None. Update D from mapping/iterable E and F.

If E present and has a .keys() method, does: for k in E: D[k] = E[k] If E present and lacks .keys() method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v

**update\_not\_jsonable** (*\*args*)

Add entries to the \_\_not\_jsonable list

**values** () → an object providing a view on D's values

## 3.9 MISPTag

**class** pymisp.MISPTag

**clear** () → None. Remove all items from D.

**edited**

Recursively check if an object has been edited and update the flag accordingly to the parent objects

**from\_json** (*json\_string*)

Load a JSON string

**get** (*k*, *d*) → *D*[*k*] if *k* in *D*, else *d*. *d* defaults to *None*.

**items** () → a set-like object providing a view on *D*'s items

**jsonable** ()

This method is used by the JSON encoder

**keys** () → a set-like object providing a view on *D*'s keys

**pop** (*k*, *d*) → *v*, remove specified key and return the corresponding value.

If key is not found, *d* is returned if given, otherwise *KeyError* is raised.

**popitem** () → (*k*, *v*), remove and return some (key, value) pair

as a 2-tuple; but raise *KeyError* if *D* is empty.

**properties**

All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the properties starting with a *\_* (private), or listed in *\_\_not\_jsonable* will be skipped.

**set\_not\_jsonable** (*\*args*)

Set *\_\_not\_jsonable* to a new list

**setdefault** (*k*, *d*) → *D*.get(*k*,*d*), also set *D*[*k*]=*d* if *k* not in *D*

**to\_dict** ()

Dump the class to a dictionary. This method automatically removes the timestamp recursively in every object that has been edited in order to let MISP update the event accordingly.

**to\_json** ()

Dump recursively any class of type *MISPAbstract* to a json string

**update** (*[E]*, *\*\*F*) → *None*. Update *D* from mapping/iterable *E* and *F*.

If *E* present and has a *.keys()* method, does: for *k* in *E*: *D*[*k*] = *E*[*k*] If *E* present and lacks *.keys()* method, does: for (*k*, *v*) in *E*: *D*[*k*] = *v* In either case, this is followed by: for *k*, *v* in *F*.items(): *D*[*k*] = *v*

**update\_not\_jsonable** (*\*args*)

Add entries to the *\_\_not\_jsonable* list

**values** () → an object providing a view on *D*'s values

## 3.10 MISPUser

```
class pymisp.MISPUser
```

**clear** () → *None*. Remove all items from *D*.

**edited**

Recursively check if an object has been edited and update the flag accordingly to the parent objects

**from\_dict** (*\*\*kwargs*)

Loading all the parameters as class properties, if they aren't *None*. This method aims to be called when all the properties requiring a special treatment are processed. Note: This method is used when you initialize an object with existing data so by default, the class is flagged as not edited.

**from\_json** (*json\_string*)

Load a JSON string

**get** (*k*, *d*) → *D*[*k*] if *k* in *D*, else *d*. *d* defaults to *None*.

**items** () → a set-like object providing a view on *D*'s items

**jsonable** ()

This method is used by the JSON encoder

**keys** () → a set-like object providing a view on *D*'s keys

**pop** (*k*, *d*) → *v*, remove specified key and return the corresponding value.

If key is not found, *d* is returned if given, otherwise *KeyError* is raised.

**popitem** () → (*k*, *v*), remove and return some (key, value) pair

as a 2-tuple; but raise *KeyError* if *D* is empty.

**properties**

All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the properties starting with a *\_* (private), or listed in *\_\_not\_jsonable* will be skipped.

**set\_not\_jsonable** (*\*args*)

Set *\_\_not\_jsonable* to a new list

**setdefault** (*k*, *d*) → *D*.*get*(*k*,*d*), also set *D*[*k*]=*d* if *k* not in *D*

**to\_dict** ()

Dump the lass to a dictionary. This method automatically removes the timestamp recursively in every object that has been edited in order to let MISP update the event accordingly.

**to\_json** ()

Dump recursively any class of type *MISPAbstract* to a json string

**update** (*[E]*, *\*\*F*) → *None*. Update *D* from mapping/iterable *E* and *F*.

If *E* present and has a *.keys()* method, does: for *k* in *E*: *D*[*k*] = *E*[*k*] If *E* present and lacks *.keys()* method, does: for (*k*, *v*) in *E*: *D*[*k*] = *v* In either case, this is followed by: for *k*, *v* in *F*.*items*(): *D*[*k*] = *v*

**update\_not\_jsonable** (*\*args*)

Add entries to the *\_\_not\_jsonable* list

**values** () → an object providing a view on *D*'s values

## 3.11 MISPOrganisation

**class** *pymisp.MISPOrganisation*

**clear** () → *None*. Remove all items from *D*.

**edited**

Recursively check if an object has been edited and update the flag accordingly to the parent objects

**from\_dict** (*\*\*kwargs*)

Loading all the parameters as class properties, if they aren't *None*. This method aims to be called when all the properties requiring a special treatment are processed. Note: This method is used when you initialize an object with existing data so by default, the class is flagged as not edited.

**from\_json** (*json\_string*)

Load a JSON string

**get** (*k*, *d*) → *D*[*k*] if *k* in *D*, else *d*. *d* defaults to *None*.

**items** () → a set-like object providing a view on D's items

**jsonable** ()

This method is used by the JSON encoder

**keys** () → a set-like object providing a view on D's keys

**pop** (*k*, *d*) → *v*, remove specified key and return the corresponding value.

If key is not found, *d* is returned if given, otherwise `KeyError` is raised.

**popitem** () → (*k*, *v*), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

**properties**

All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the properties starting with a `_` (private), or listed in `__not_jsonable` will be skipped.

**set\_not\_jsonable** (*\*args*)

Set `__not_jsonable` to a new list

**setdefault** (*k*, *d*) → *D.get(k,d)*, also set *D[k]=d* if *k* not in *D*

**to\_dict** ()

Dump the class to a dictionary. This method automatically removes the timestamp recursively in every object that has been edited in order to let MISP update the event accordingly.

**to\_json** ()

Dump recursively any class of type `MISPAbstract` to a json string

**update** (*[E]*, *\*\*F*) → `None`. Update *D* from mapping/iterable *E* and *F*.

If *E* present and has a `.keys()` method, does: for *k* in *E*: *D[k] = E[k]* If *E* present and lacks `.keys()` method, does: for (*k*, *v*) in *E*: *D[k] = v* In either case, this is followed by: for *k*, *v* in *F.items()*: *D[k] = v*

**update\_not\_jsonable** (*\*args*)

Add entries to the `__not_jsonable` list

**values** () → an object providing a view on D's values

## 4.1 File Object

```
class pymisp.tools.FileObject (filepath=None, pseudofile=None, filename=None, standalone=True, **kwargs)
```

```
add_attribute (object_relation, **value)
```

Add an attribute. *object\_relation* is required and the value key is a dictionary with all the keys supported by MISPAtribute

```
add_reference (referenced_uuid, relationship_type, comment=None, **kwargs)
```

Add a link (uuid) to an other object

```
clear () → None. Remove all items from D.
```

```
edited
```

Recursively check if an object has been edited and update the flag accordingly to the parent objects

```
from_json (json_string)
```

Load a JSON string

```
get (k[, d]) → D[k] if k in D, else d. d defaults to None.
```

```
get_attributes_by_relation (object_relation)
```

Returns the list of attributes with the given object relation in the object

```
has_attributes_by_relation (list_of_relations)
```

True if all the relations in the list are defined in the object

```
items () → a set-like object providing a view on D's items
```

```
jsonable ()
```

This method is used by the JSON encoder

```
keys () → a set-like object providing a view on D's keys
```

```
pop (k[, d]) → v, remove specified key and return the corresponding value.
```

If key is not found, d is returned if given, otherwise `KeyError` is raised.

**popitem** () → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

#### **properties**

All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the properties starting with a `_` (private), or listed in `__not_jsonable` will be skipped.

**set\_not\_jsonable** (\*args)  
Set `__not_jsonable` to a new list

**setdefault** (k[, d]) → D.get(k,d), also set D[k]=d if k not in D

**update** ([E], \*\*F) → None. Update D from mapping/iterable E and F.  
If E present and has a `.keys()` method, does: for k in E: D[k] = E[k] If E present and lacks `.keys()` method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v

**update\_not\_jsonable** (\*args)  
Add entries to the `__not_jsonable` list

**values** () → an object providing a view on D's values

## 4.2 ELF Object

**class** `pymisp.tools.ELFObject` (*parsed=None, filepath=None, pseudofile=None, standalone=True, \*\*kwargs*)

**add\_attribute** (*object\_relation, \*\*value*)  
Add an attribute. *object\_relation* is required and the value key is a dictionary with all the keys supported by `MISPAttribute`

**add\_reference** (*referenced\_uuid, relationship\_type, comment=None, \*\*kwargs*)  
Add a link (uuid) to an other object

**clear** () → None. Remove all items from D.

#### **edited**

Recursively check if an object has been edited and update the flag accordingly to the parent objects

**from\_json** (*json\_string*)  
Load a JSON string

**get** (k[, d]) → D[k] if k in D, else d. d defaults to None.

**get\_attributes\_by\_relation** (*object\_relation*)  
Returns the list of attributes with the given object relation in the object

**has\_attributes\_by\_relation** (*list\_of\_relations*)  
True if all the relations in the list are defined in the object

**items** () → a set-like object providing a view on D's items

**jsonable** ()  
This method is used by the JSON encoder

**keys** () → a set-like object providing a view on D's keys

**pop** (k[, d]) → v, remove specified key and return the corresponding value.  
If key is not found, d is returned if given, otherwise `KeyError` is raised.

**popitem** () → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.



**properties**

All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the properties starting with a `_` (private), or listed in `__not_jsonable` will be skipped.

**set\_not\_jsonable** (\*args)

Set `__not_jsonable` to a new list

**setdefault** (k[, d]) → D.get(k,d), also set D[k]=d if k not in D

**update** ([E], \*\*F) → None. Update D from mapping/iterable E and F.

If E present and has a `.keys()` method, does: for k in E: D[k] = E[k] If E present and lacks `.keys()` method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v

**update\_not\_jsonable** (\*args)

Add entries to the `__not_jsonable` list

**values** () → an object providing a view on D's values

**class** pymisp.tools.ELFSectionObject (section, standalone=True, \*\*kwargs)

**add\_attribute** (object\_relation, \*\*value)

Add an attribute. `object_relation` is required and the value key is a dictionary with all the keys supported by MISPAtribute

**add\_reference** (referenced\_uuid, relationship\_type, comment=None, \*\*kwargs)

Add a link (uuid) to an other object

**clear** () → None. Remove all items from D.

**edited**

Recursively check if an object has been edited and update the flag accordingly to the parent objects

**from\_json** (json\_string)

Load a JSON string

**get** (k[, d]) → D[k] if k in D, else d. d defaults to None.

**get\_attributes\_by\_relation** (object\_relation)

Returns the list of attributes with the given object relation in the object

**has\_attributes\_by\_relation** (list\_of\_relations)

True if all the relations in the list are defined in the object

**items** () → a set-like object providing a view on D's items

**jsonable** ()

This method is used by the JSON encoder

**keys** () → a set-like object providing a view on D's keys

**pop** (k[, d]) → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise `KeyError` is raised.

**popitem** () → (k, v), remove and return some (key, value) pair

as a 2-tuple; but raise `KeyError` if D is empty.

**properties**

All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the properties starting with a `_` (private), or listed in `__not_jsonable` will be skipped.

**set\_not\_jsonable** (\*args)

Set `__not_jsonable` to a new list

**setdefault** (k[, d]) → D.get(k,d), also set D[k]=d if k not in D

**update** (*[E]*, *\*\*F*) → None. Update D from mapping/iterable E and F.  
If E present and has a .keys() method, does: for k in E: D[k] = E[k] If E present and lacks .keys() method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v

**update\_not\_jsonable** (*\*args*)  
Add entries to the \_\_not\_jsonable list

**values** () → an object providing a view on D's values

## 4.3 PE Object

**class** pymisp.tools.PEObject (*parsed=None, filepath=None, pseudofile=None, standalone=True, \*\*kwargs*)

**add\_attribute** (*object\_relation, \*\*value*)  
Add an attribute. object\_relation is required and the value key is a dictionary with all the keys supported by MISPAtribute

**add\_reference** (*referenced\_uuid, relationship\_type, comment=None, \*\*kwargs*)  
Add a link (uuid) to an other object

**clear** () → None. Remove all items from D.

**edited**  
Recursively check if an object has been edited and update the flag accordingly to the parent objects

**from\_json** (*json\_string*)  
Load a JSON string

**get** (*k[, d]*) → D[k] if k in D, else d. d defaults to None.

**get\_attributes\_by\_relation** (*object\_relation*)  
Returns the list of attributes with the given object relation in the object

**has\_attributes\_by\_relation** (*list\_of\_relations*)  
True if all the relations in the list are defined in the object

**items** () → a set-like object providing a view on D's items

**jsonable** ()  
This method is used by the JSON encoder

**keys** () → a set-like object providing a view on D's keys

**pop** (*k[, d]*) → v, remove specified key and return the corresponding value.  
If key is not found, d is returned if given, otherwise KeyError is raised.

**popitem** () → (k, v), remove and return some (key, value) pair  
as a 2-tuple; but raise KeyError if D is empty.

**properties**  
All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the properties starting with a \_ (private), or listed in \_\_not\_jsonable will be skipped.

**set\_not\_jsonable** (*\*args*)  
Set \_\_not\_jsonable to a new list

**setdefault** (*k[, d]*) → D.get(k,d), also set D[k]=d if k not in D

**update** (*[E]*, *\*\*F*) → None. Update D from mapping/iterable E and F.  
 If E present and has a .keys() method, does: for k in E: D[k] = E[k] If E present and lacks .keys() method,  
 does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v

**update\_not\_jsonable** (*\*args*)  
 Add entries to the \_\_not\_jsonable list

**values** () → an object providing a view on D's values

**class** pymisp.tools.**PESectionObject** (*section*, *standalone=True*, *\*\*kwargs*)

**add\_attribute** (*object\_relation*, *\*\*value*)  
 Add an attribute. object\_relation is required and the value key is a dictionary with all the keys supported  
 by MISPAtribute

**add\_reference** (*referenced\_uuid*, *relationship\_type*, *comment=None*, *\*\*kwargs*)  
 Add a link (uuid) to an other object

**clear** () → None. Remove all items from D.

**edited**  
 Recursively check if an object has been edited and update the flag accordingly to the parent objects

**from\_json** (*json\_string*)  
 Load a JSON string

**get** (*k*, *d*) → D[k] if k in D, else d. d defaults to None.

**get\_attributes\_by\_relation** (*object\_relation*)  
 Returns the list of attributes with the given object relation in the object

**has\_attributes\_by\_relation** (*list\_of\_relations*)  
 True if all the relations in the list are defined in the object

**items** () → a set-like object providing a view on D's items

**jsonable** ()  
 This method is used by the JSON encoder

**keys** () → a set-like object providing a view on D's keys

**pop** (*k*, *d*) → v, remove specified key and return the corresponding value.  
 If key is not found, d is returned if given, otherwise KeyError is raised.

**popitem** () → (k, v), remove and return some (key, value) pair  
 as a 2-tuple; but raise KeyError if D is empty.

**properties**  
 All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the  
 properties starting with a \_ (private), or listed in \_\_not\_jsonable will be skipped.

**set\_not\_jsonable** (*\*args*)  
 Set \_\_not\_jsonable to a new list

**setdefault** (*k*, *d*) → D.get(k,d), also set D[k]=d if k not in D

**update** (*[E]*, *\*\*F*) → None. Update D from mapping/iterable E and F.  
 If E present and has a .keys() method, does: for k in E: D[k] = E[k] If E present and lacks .keys() method,  
 does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v

**update\_not\_jsonable** (*\*args*)  
 Add entries to the \_\_not\_jsonable list

**values** () → an object providing a view on D's values

## 4.4 Mach-O Object

```
class pymisp.tools.MachOObject (parsed=None, filepath=None, pseudofile=None, standalone=True, **kwargs)

add_attribute (object_relation, **value)
    Add an attribute. object_relation is required and the value key is a dictionary with all the keys supported
    by MISPAtribute

add_reference (referenced_uuid, relationship_type, comment=None, **kwargs)
    Add a link (uuid) to an other object

clear () → None. Remove all items from D.

edited
    Recursively check if an object has been edited and update the flag accordingly to the parent objects

from_json (json_string)
    Load a JSON string

get (k[, d]) → D[k] if k in D, else d. d defaults to None.

get_attributes_by_relation (object_relation)
    Returns the list of attributes with the given object relation in the object

has_attributes_by_relation (list_of_relations)
    True if all the relations in the list are defined in the object

items () → a set-like object providing a view on D's items

jsonable ()
    This method is used by the JSON encoder

keys () → a set-like object providing a view on D's keys

pop (k[, d]) → v, remove specified key and return the corresponding value.
    If key is not found, d is returned if given, otherwise KeyError is raised.

popitem () → (k, v), remove and return some (key, value) pair
    as a 2-tuple; but raise KeyError if D is empty.

properties
    All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the
    properties starting with a _ (private), or listed in __not_jsonable will be skipped.

set_not_jsonable (*args)
    Set __not_jsonable to a new list

setdefault (k[, d]) → D.get(k,d), also set D[k]=d if k not in D

update ([E], **F) → None. Update D from mapping/iterable E and F.
    If E present and has a .keys() method, does: for k in E: D[k] = E[k] If E present and lacks .keys() method,
    does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v

update_not_jsonable (*args)
    Add entries to the __not_jsonable list

values () → an object providing a view on D's values

class pymisp.tools.MachOSectionObject (section, standalone=True, **kwargs)
```

**add\_attribute** (*object\_relation*, *\*\*value*)  
 Add an attribute. *object\_relation* is required and the value key is a dictionary with all the keys supported by MISPAtribute

**add\_reference** (*referenced\_uuid*, *relationship\_type*, *comment=None*, *\*\*kwargs*)  
 Add a link (uuid) to an other object

**clear** () → None. Remove all items from D.

**edited**  
 Recursively check if an object has been edited and update the flag accordingly to the parent objects

**from\_json** (*json\_string*)  
 Load a JSON string

**get** (*k*, *d*) → D[k] if k in D, else d. d defaults to None.

**get\_attributes\_by\_relation** (*object\_relation*)  
 Returns the list of attributes with the given object relation in the object

**has\_attributes\_by\_relation** (*list\_of\_relations*)  
 True if all the relations in the list are defined in the object

**items** () → a set-like object providing a view on D's items

**jsonable** ()  
 This method is used by the JSON encoder

**keys** () → a set-like object providing a view on D's keys

**pop** (*k*, *d*) → v, remove specified key and return the corresponding value.  
 If key is not found, d is returned if given, otherwise KeyError is raised.

**popitem** () → (k, v), remove and return some (key, value) pair  
 as a 2-tuple; but raise KeyError if D is empty.

**properties**  
 All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the properties starting with a \_ (private), or listed in `__not_jsonable` will be skipped.

**set\_not\_jsonable** (*\*args*)  
 Set `__not_jsonable` to a new list

**setdefault** (*k*, *d*) → D.get(k,d), also set D[k]=d if k not in D

**update** (*[E]*, *\*\*F*) → None. Update D from mapping/iterable E and F.  
 If E present and has a .keys() method, does: for k in E: D[k] = E[k] If E present and lacks .keys() method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v

**update\_not\_jsonable** (*\*args*)  
 Add entries to the `__not_jsonable` list

**values** () → an object providing a view on D's values

## 4.5 VT Report Object

```
class pymisp.tools.VTReportObject (apikey, indicator, vt_proxies=None, standalone=True,
                                   **kwargs)
```

VirusTotal Report

**ApiKey** VirusTotal API key (private works, but only public features are supported right now)

**Indicator** IOC to search VirusTotal for

**add\_attribute** (*object\_relation*, *\*\*value*)

Add an attribute. *object\_relation* is required and the value key is a dictionary with all the keys supported by MISPAtribute

**add\_reference** (*referenced\_uuid*, *relationship\_type*, *comment=None*, *\*\*kwargs*)

Add a link (uuid) to an other object

**clear** () → None. Remove all items from D.

**edited**

Recursively check if an object has been edited and update the flag accordingly to the parent objects

**from\_json** (*json\_string*)

Load a JSON string

**generate\_attributes** ()

Parse the VirusTotal report for relevant attributes

**get** (*k*, *d*) → D[k] if k in D, else d. d defaults to None.

**get\_attributes\_by\_relation** (*object\_relation*)

Returns the list of attributes with the given object relation in the object

**has\_attributes\_by\_relation** (*list\_of\_relations*)

True if all the relations in the list are defined in the object

**items** () → a set-like object providing a view on D's items

**jsonable** ()

This method is used by the JSON encoder

**keys** () → a set-like object providing a view on D's keys

**pop** (*k*, *d*) → v, remove specified key and return the corresponding value.

If key is not found, d is returned if given, otherwise KeyError is raised.

**popitem** () → (k, v), remove and return some (key, value) pair

as a 2-tuple; but raise KeyError if D is empty.

**properties**

All the class public properties that will be dumped in the dictionary, and the JSON export. Note: all the properties starting with a \_ (private), or listed in `__not_jsonable` will be skipped.

**set\_not\_jsonable** (*\*args*)

Set `__not_jsonable` to a new list

**setdefault** (*k*, *d*) → D.get(k,d), also set D[k]=d if k not in D

**update** (*[E]*, *\*\*F*) → None. Update D from mapping/iterable E and F.

If E present and has a `.keys()` method, does: for k in E: D[k] = E[k] If E present and lacks `.keys()` method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k, v in F.items(): D[k] = v

**update\_not\_jsonable** (*\*args*)

Add entries to the `__not_jsonable` list

**values** () → an object providing a view on D's values

## 4.6 STIX

`pymisp.tools.stix.load_stix` (*stix, distribution=3, threat\_level\_id=2, analysis=0*)

Returns a MISPEvent object from a STIX package

`pymisp.tools.stix.make_stix_package` (*misp\_event, to\_json=False, to\_xml=False*)

Returns a STIXPackage from a MISPEvent.

Optionally can return the package in json or xml.

## 4.7 OpenIOC

`tools.load_openioc` (*openioc*)

`tools.load_openioc_file` (*openioc\_path*)





## CHAPTER 5

---

### Indices and tables

---

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



**p**

pymisp, 9  
pymisp.tools, 27  
pymisp.tools.stix, 35



## A

- AbstractMISP (class in pymisp), 17
- add\_attachment() (pymisp.PyMISP method), 9
- add\_attribute() (pymisp.MISPEvent method), 18
- add\_attribute() (pymisp.MISPObject method), 21
- add\_attribute() (pymisp.tools.ELFObject method), 28
- add\_attribute() (pymisp.tools.ELFSectionObject method), 29
- add\_attribute() (pymisp.tools.FileObject method), 27
- add\_attribute() (pymisp.tools.MachOObject method), 32
- add\_attribute() (pymisp.tools.MachOSectionObject method), 32
- add\_attribute() (pymisp.tools.PEObject method), 30
- add\_attribute() (pymisp.tools.PESectionObject method), 31
- add\_attribute() (pymisp.tools.VTReportObject method), 34
- add\_attribute\_tag() (pymisp.MISPEvent method), 18
- add\_detection\_name() (pymisp.PyMISP method), 10
- add\_domain() (pymisp.PyMISP method), 10
- add\_domain\_ip() (pymisp.PyMISP method), 10
- add\_domains\_ips() (pymisp.PyMISP method), 10
- add\_email\_attachment() (pymisp.PyMISP method), 10
- add\_email\_dst() (pymisp.PyMISP method), 10
- add\_email\_src() (pymisp.PyMISP method), 10
- add\_email\_subject() (pymisp.PyMISP method), 10
- add\_event() (pymisp.PyMISP method), 10
- add\_filename() (pymisp.PyMISP method), 10
- add\_hashes() (pymisp.PyMISP method), 10
- add\_hostname() (pymisp.PyMISP method), 10
- add\_internal\_comment() (pymisp.PyMISP method), 10
- add\_internal\_link() (pymisp.PyMISP method), 10
- add\_internal\_other() (pymisp.PyMISP method), 10
- add\_internal\_text() (pymisp.PyMISP method), 11
- add\_ipdst() (pymisp.PyMISP method), 11
- add\_ipsrc() (pymisp.PyMISP method), 11
- add\_mutex() (pymisp.PyMISP method), 11
- add\_named\_attribute() (pymisp.PyMISP method), 11
- add\_net\_other() (pymisp.PyMISP method), 11
- add\_object() (pymisp.MISPEvent method), 18
- add\_object() (pymisp.PyMISP method), 11
- add\_object\_reference() (pymisp.PyMISP method), 11
- add\_pattern() (pymisp.PyMISP method), 11
- add\_pipe() (pymisp.PyMISP method), 11
- add\_proposal() (pymisp.MISPAAttribute method), 20
- add\_proposal() (pymisp.MISPEvent method), 18
- add\_proposal() (pymisp.MISPObjectAttribute method), 22
- add\_reference() (pymisp.MISPObject method), 21
- add\_reference() (pymisp.tools.ELFObject method), 28
- add\_reference() (pymisp.tools.ELFSectionObject method), 29
- add\_reference() (pymisp.tools.FileObject method), 27
- add\_reference() (pymisp.tools.MachOObject method), 32
- add\_reference() (pymisp.tools.MachOSectionObject method), 33
- add\_reference() (pymisp.tools.PEObject method), 30
- add\_reference() (pymisp.tools.PESectionObject method), 31
- add\_reference() (pymisp.tools.VTReportObject method), 34
- add\_regkey() (pymisp.PyMISP method), 11
- add\_regkeys() (pymisp.PyMISP method), 11
- add\_shadow\_attribute() (pymisp.MISPAAttribute method), 20
- add\_shadow\_attribute() (pymisp.MISPEvent method), 18
- add\_shadow\_attribute() (pymisp.MISPObjectAttribute method), 22
- add\_snort() (pymisp.PyMISP method), 11
- add\_target\_email() (pymisp.PyMISP method), 11
- add\_target\_external() (pymisp.PyMISP method), 11
- add\_target\_location() (pymisp.PyMISP method), 11
- add\_target\_machine() (pymisp.PyMISP method), 12
- add\_target\_org() (pymisp.PyMISP method), 12
- add\_target\_user() (pymisp.PyMISP method), 12
- add\_threat\_actor() (pymisp.PyMISP method), 12
- add\_traffic\_pattern() (pymisp.PyMISP method), 12
- add\_url() (pymisp.PyMISP method), 12
- add\_useragent() (pymisp.PyMISP method), 12

add\_yara() (pymisp.PyMISP method), 12  
 av\_detection\_link() (pymisp.PyMISP method), 12

## C

cache\_all\_feeds() (pymisp.PyMISP method), 12  
 cache\_feed() (pymisp.PyMISP method), 12  
 cache\_feeds\_all() (pymisp.PyMISP method), 12  
 cache\_feeds\_freetext() (pymisp.PyMISP method), 12  
 cache\_feeds\_misp() (pymisp.PyMISP method), 12  
 change\_comment() (pymisp.PyMISP method), 12  
 change\_sharing\_group() (pymisp.PyMISP method), 12  
 change\_threat\_level() (pymisp.PyMISP method), 12  
 change\_toids() (pymisp.PyMISP method), 12  
 clear() (pymisp.MISPAttribute method), 20  
 clear() (pymisp.MISPEvent method), 18  
 clear() (pymisp.MISPObject method), 21  
 clear() (pymisp.MISPObjectAttribute method), 22  
 clear() (pymisp.MISPObjectReference method), 23  
 clear() (pymisp.MISPOrganisation method), 25  
 clear() (pymisp.MISPTag method), 23  
 clear() (pymisp.MISPUser method), 24  
 clear() (pymisp.tools.ELFObject method), 28  
 clear() (pymisp.tools.ELFSectionObject method), 29  
 clear() (pymisp.tools.FileObject method), 27  
 clear() (pymisp.tools.MachOObject method), 32  
 clear() (pymisp.tools.MachOSectionObject method), 33  
 clear() (pymisp.tools.PEObject method), 30  
 clear() (pymisp.tools.PESectionObject method), 31  
 clear() (pymisp.tools.VTReportObject method), 34  
 compare\_feeds() (pymisp.PyMISP method), 12

## D

delete() (pymisp.MISPAttribute method), 20  
 delete() (pymisp.MISPObjectAttribute method), 22  
 delete\_attribute() (pymisp.MISPEvent method), 18  
 delete\_attribute() (pymisp.PyMISP method), 13  
 delete\_event() (pymisp.PyMISP method), 13  
 delete\_object() (pymisp.PyMISP method), 13  
 delete\_object\_reference() (pymisp.PyMISP method), 13  
 deprecated() (in module pymisp), 9  
 download\_all\_suricata() (pymisp.PyMISP method), 13  
 download\_last() (pymisp.PyMISP method), 13  
 download\_samples() (pymisp.PyMISP method), 13  
 download\_suricata\_rule\_event() (pymisp.PyMISP method), 13

## E

edited (pymisp.AbstractMISP attribute), 17  
 edited (pymisp.MISPAttribute attribute), 20  
 edited (pymisp.MISPEvent attribute), 18  
 edited (pymisp.MISPObject attribute), 21  
 edited (pymisp.MISPObjectAttribute attribute), 22  
 edited (pymisp.MISPObjectReference attribute), 23  
 edited (pymisp.MISPOrganisation attribute), 25

edited (pymisp.MISPTag attribute), 23  
 edited (pymisp.MISPUser attribute), 24  
 edited (pymisp.tools.ELFObject attribute), 28  
 edited (pymisp.tools.ELFSectionObject attribute), 29  
 edited (pymisp.tools.FileObject attribute), 27  
 edited (pymisp.tools.MachOObject attribute), 32  
 edited (pymisp.tools.MachOSectionObject attribute), 33  
 edited (pymisp.tools.PEObject attribute), 30  
 edited (pymisp.tools.PESectionObject attribute), 31  
 edited (pymisp.tools.VTReportObject attribute), 34  
 ELFObject (class in pymisp.tools), 28  
 ELFSectionObject (class in pymisp.tools), 29

## F

fast\_publish() (pymisp.PyMISP method), 13  
 fetch\_feed() (pymisp.PyMISP method), 13  
 FileObject (class in pymisp.tools), 27  
 flatten\_error\_messages() (pymisp.PyMISP method), 13  
 freetext() (pymisp.PyMISP method), 13  
 from\_dict() (pymisp.AbstractMISP method), 17  
 from\_dict() (pymisp.MISPOrganisation method), 25  
 from\_dict() (pymisp.MISPUser method), 24  
 from\_json() (pymisp.AbstractMISP method), 17  
 from\_json() (pymisp.MISPAttribute method), 20  
 from\_json() (pymisp.MISPEvent method), 18  
 from\_json() (pymisp.MISPObject method), 21  
 from\_json() (pymisp.MISPObjectAttribute method), 22  
 from\_json() (pymisp.MISPObjectReference method), 23  
 from\_json() (pymisp.MISPOrganisation method), 25  
 from\_json() (pymisp.MISPTag method), 24  
 from\_json() (pymisp.MISPUser method), 24  
 from\_json() (pymisp.tools.ELFObject method), 28  
 from\_json() (pymisp.tools.ELFSectionObject method), 29  
 from\_json() (pymisp.tools.FileObject method), 27  
 from\_json() (pymisp.tools.MachOObject method), 32  
 from\_json() (pymisp.tools.MachOSectionObject method), 33  
 from\_json() (pymisp.tools.PEObject method), 30  
 from\_json() (pymisp.tools.PESectionObject method), 31  
 from\_json() (pymisp.tools.VTReportObject method), 34

## G

generate\_attributes() (pymisp.tools.VTReportObject method), 34  
 get() (pymisp.MISPAttribute method), 20  
 get() (pymisp.MISPEvent method), 19  
 get() (pymisp.MISPObject method), 21  
 get() (pymisp.MISPObjectAttribute method), 22  
 get() (pymisp.MISPObjectReference method), 23  
 get() (pymisp.MISPOrganisation method), 25  
 get() (pymisp.MISPTag method), 24  
 get() (pymisp.MISPUser method), 25  
 get() (pymisp.PyMISP method), 13

- get() (pymisp.tools.ELFObject method), 28  
 get() (pymisp.tools.ELFSectionObject method), 29  
 get() (pymisp.tools.FileObject method), 27  
 get() (pymisp.tools.MachOObject method), 32  
 get() (pymisp.tools.MachOSectionObject method), 33  
 get() (pymisp.tools.PEObject method), 30  
 get() (pymisp.tools.PESectionObject method), 31  
 get() (pymisp.tools.VTReportObject method), 34  
 get\_all\_attributes\_txt() (pymisp.PyMISP method), 13  
 get\_all\_tags() (pymisp.PyMISP method), 13  
 get\_api\_version() (pymisp.PyMISP method), 13  
 get\_api\_version\_master() (pymisp.PyMISP method), 13  
 get\_attachment() (pymisp.PyMISP method), 13  
 get\_attribute\_tag() (pymisp.MISPEvent method), 19  
 get\_attributes\_by\_relation() (pymisp.MISPObject method), 21  
 get\_attributes\_by\_relation() (pymisp.tools.ELFObject method), 28  
 get\_attributes\_by\_relation() (pymisp.tools.ELFSectionObject method), 29  
 get\_attributes\_by\_relation() (pymisp.tools.FileObject method), 27  
 get\_attributes\_by\_relation() (pymisp.tools.MachOObject method), 32  
 get\_attributes\_by\_relation() (pymisp.tools.MachOSectionObject method), 33  
 get\_attributes\_by\_relation() (pymisp.tools.PEObject method), 30  
 get\_attributes\_by\_relation() (pymisp.tools.PESectionObject method), 31  
 get\_attributes\_by\_relation() (pymisp.tools.VTReportObject method), 34  
 get\_attributes\_statistics() (pymisp.PyMISP method), 14  
 get\_csv() (pymisp.PyMISP method), 14  
 get\_event() (pymisp.PyMISP method), 14  
 get\_events\_last\_modified() (pymisp.PyMISP method), 14  
 get\_index() (pymisp.PyMISP method), 14  
 get\_live\_query\_acl() (pymisp.PyMISP method), 14  
 get\_object\_by\_id() (pymisp.MISPEvent method), 19  
 get\_object\_template\_id() (pymisp.PyMISP method), 14  
 get\_object\_templates\_list() (pymisp.PyMISP method), 14  
 get\_recommended\_api\_version() (pymisp.PyMISP method), 14  
 get\_roles\_list() (pymisp.PyMISP method), 14  
 get\_sharing\_groups() (pymisp.PyMISP method), 14  
 get\_stix\_event() (pymisp.PyMISP method), 14  
 get\_tags\_list() (pymisp.PyMISP method), 14  
 get\_tags\_statistics() (pymisp.PyMISP method), 14  
 get\_version() (pymisp.PyMISP method), 15  
 get\_version\_master() (pymisp.PyMISP method), 15  
 get\_yara() (pymisp.PyMISP method), 15
- ## H
- has\_attributes\_by\_relation() (pymisp.MISPObject method), 21  
 has\_attributes\_by\_relation() (pymisp.tools.ELFObject method), 28  
 has\_attributes\_by\_relation() (pymisp.tools.ELFSectionObject method), 29  
 has\_attributes\_by\_relation() (pymisp.tools.FileObject method), 27  
 has\_attributes\_by\_relation() (pymisp.tools.MachOObject method), 32  
 has\_attributes\_by\_relation() (pymisp.tools.MachOSectionObject method), 33  
 has\_attributes\_by\_relation() (pymisp.tools.PEObject method), 30  
 has\_attributes\_by\_relation() (pymisp.tools.PESectionObject method), 31  
 has\_attributes\_by\_relation() (pymisp.tools.VTReportObject method), 34
- ## I
- items() (pymisp.MISPAttribute method), 20  
 items() (pymisp.MISPEvent method), 19  
 items() (pymisp.MISPObject method), 21  
 items() (pymisp.MISPObjectAttribute method), 22  
 items() (pymisp.MISPObjectReference method), 23  
 items() (pymisp.MISPOrganisation method), 25  
 items() (pymisp.MISPTag method), 24  
 items() (pymisp.MISPUser method), 25  
 items() (pymisp.tools.ELFObject method), 28  
 items() (pymisp.tools.ELFSectionObject method), 29  
 items() (pymisp.tools.FileObject method), 27  
 items() (pymisp.tools.MachOObject method), 32  
 items() (pymisp.tools.MachOSectionObject method), 33  
 items() (pymisp.tools.PEObject method), 30  
 items() (pymisp.tools.PESectionObject method), 31  
 items() (pymisp.tools.VTReportObject method), 34
- ## J
- jsonable() (pymisp.AbstractMISP method), 18  
 jsonable() (pymisp.MISPAttribute method), 20  
 jsonable() (pymisp.MISPEvent method), 19  
 jsonable() (pymisp.MISPObject method), 21  
 jsonable() (pymisp.MISPObjectAttribute method), 22  
 jsonable() (pymisp.MISPObjectReference method), 23  
 jsonable() (pymisp.MISPOrganisation method), 26  
 jsonable() (pymisp.MISPTag method), 24  
 jsonable() (pymisp.MISPUser method), 25

jsonable() (pymisp.tools.ELFObject method), 28  
 jsonable() (pymisp.tools.ELFSectionObject method), 29  
 jsonable() (pymisp.tools.FileObject method), 27  
 jsonable() (pymisp.tools.MachOObject method), 32  
 jsonable() (pymisp.tools.MachOSectionObject method), 33  
 jsonable() (pymisp.tools.PEObject method), 30  
 jsonable() (pymisp.tools.PESectionObject method), 31  
 jsonable() (pymisp.tools.VTReportObject method), 34

## K

keys() (pymisp.MISPAttribute method), 20  
 keys() (pymisp.MISPEvent method), 19  
 keys() (pymisp.MISPObject method), 21  
 keys() (pymisp.MISPObjectAttribute method), 22  
 keys() (pymisp.MISPObjectReference method), 23  
 keys() (pymisp.MISPOrganisation method), 26  
 keys() (pymisp.MISPTag method), 24  
 keys() (pymisp.MISPUser method), 25  
 keys() (pymisp.tools.ELFObject method), 28  
 keys() (pymisp.tools.ELFSectionObject method), 29  
 keys() (pymisp.tools.FileObject method), 27  
 keys() (pymisp.tools.MachOObject method), 32  
 keys() (pymisp.tools.MachOSectionObject method), 33  
 keys() (pymisp.tools.PEObject method), 30  
 keys() (pymisp.tools.PESectionObject method), 31  
 keys() (pymisp.tools.VTReportObject method), 34  
 known\_types (pymisp.MISPAttribute attribute), 20  
 known\_types (pymisp.MISPObjectAttribute attribute), 22

## L

load() (pymisp.MISPEvent method), 19  
 load\_file() (pymisp.MISPEvent method), 19  
 load\_openioc() (pymisp.tools method), 35  
 load\_openioc\_file() (pymisp.tools method), 35  
 load\_stix() (in module pymisp.tools.stix), 35

## M

MachOObject (class in pymisp.tools), 32  
 MachOSectionObject (class in pymisp.tools), 32  
 make\_stix\_package() (in module pymisp.tools.stix), 35  
 malware\_binary (pymisp.MISPAttribute attribute), 20  
 malware\_binary (pymisp.MISPObjectAttribute attribute), 22  
 MISPAttribute (class in pymisp), 20  
 MISPEncode (class in pymisp), 18  
 MISPEvent (class in pymisp), 18  
 MISPObject (class in pymisp), 21  
 MISPObjectAttribute (class in pymisp), 22  
 MISPObjectReference (class in pymisp), 23  
 MISPOrganisation (class in pymisp), 25  
 MISPTag (class in pymisp), 23  
 MISPUser (class in pymisp), 24

## N

new\_event() (pymisp.PyMISP method), 15  
 new\_tag() (pymisp.PyMISP method), 15

## P

PEObject (class in pymisp.tools), 30  
 PESectionObject (class in pymisp.tools), 31  
 pop() (pymisp.MISPAttribute method), 20  
 pop() (pymisp.MISPEvent method), 19  
 pop() (pymisp.MISPObject method), 21  
 pop() (pymisp.MISPObjectAttribute method), 22  
 pop() (pymisp.MISPObjectReference method), 23  
 pop() (pymisp.MISPOrganisation method), 26  
 pop() (pymisp.MISPTag method), 24  
 pop() (pymisp.MISPUser method), 25  
 pop() (pymisp.tools.ELFObject method), 28  
 pop() (pymisp.tools.ELFSectionObject method), 29  
 pop() (pymisp.tools.FileObject method), 27  
 pop() (pymisp.tools.MachOObject method), 32  
 pop() (pymisp.tools.MachOSectionObject method), 33  
 pop() (pymisp.tools.PEObject method), 30  
 pop() (pymisp.tools.PESectionObject method), 31  
 pop() (pymisp.tools.VTReportObject method), 34  
 popitem() (pymisp.MISPAttribute method), 20  
 popitem() (pymisp.MISPEvent method), 19  
 popitem() (pymisp.MISPObject method), 21  
 popitem() (pymisp.MISPObjectAttribute method), 22  
 popitem() (pymisp.MISPObjectReference method), 23  
 popitem() (pymisp.MISPOrganisation method), 26  
 popitem() (pymisp.MISPTag method), 24  
 popitem() (pymisp.MISPUser method), 25  
 popitem() (pymisp.tools.ELFObject method), 28  
 popitem() (pymisp.tools.ELFSectionObject method), 29  
 popitem() (pymisp.tools.FileObject method), 27  
 popitem() (pymisp.tools.MachOObject method), 32  
 popitem() (pymisp.tools.MachOSectionObject method), 33  
 popitem() (pymisp.tools.PEObject method), 30  
 popitem() (pymisp.tools.PESectionObject method), 31  
 popitem() (pymisp.tools.VTReportObject method), 34  
 properties (pymisp.AbstractMISP attribute), 18  
 properties (pymisp.MISPAttribute attribute), 20  
 properties (pymisp.MISPEvent attribute), 19  
 properties (pymisp.MISPObject attribute), 21  
 properties (pymisp.MISPObjectAttribute attribute), 22  
 properties (pymisp.MISPObjectReference attribute), 23  
 properties (pymisp.MISPOrganisation attribute), 26  
 properties (pymisp.MISPTag attribute), 24  
 properties (pymisp.MISPUser attribute), 25  
 properties (pymisp.tools.ELFObject attribute), 28  
 properties (pymisp.tools.ELFSectionObject attribute), 29  
 properties (pymisp.tools.FileObject attribute), 28  
 properties (pymisp.tools.MachOObject attribute), 32



properties (pymisp.tools.MachOSectionObject attribute), 33  
 properties (pymisp.tools.PEObject attribute), 30  
 properties (pymisp.tools.PESectionObject attribute), 31  
 properties (pymisp.tools.VTReportObject attribute), 34  
 proposal\_accept() (pymisp.PyMISP method), 15  
 proposal\_add() (pymisp.PyMISP method), 15  
 proposal\_discard() (pymisp.PyMISP method), 15  
 proposal\_edit() (pymisp.PyMISP method), 15  
 proposal\_view() (pymisp.PyMISP method), 15  
 publish() (pymisp.MISPEvent method), 19  
 publish() (pymisp.PyMISP method), 15  
 pushEventToZMQ() (pymisp.PyMISP method), 15  
 PyMISP (class in pymisp), 9  
 pymisp (module), 9  
 pymisp.tools (module), 27  
 pymisp.tools.stix (module), 35

## S

search() (pymisp.PyMISP method), 15  
 search\_all() (pymisp.PyMISP method), 16  
 search\_index() (pymisp.PyMISP method), 16  
 set\_date() (pymisp.MISPEvent method), 19  
 set\_not\_jsonable() (pymisp.AbstractMISP method), 18  
 set\_not\_jsonable() (pymisp.MISPAttribute method), 20  
 set\_not\_jsonable() (pymisp.MISPEvent method), 19  
 set\_not\_jsonable() (pymisp.MISPObject method), 21  
 set\_not\_jsonable() (pymisp.MISPObjectAttribute method), 22  
 set\_not\_jsonable() (pymisp.MISPObjectReference method), 23  
 set\_not\_jsonable() (pymisp.MISPOrganisation method), 26  
 set\_not\_jsonable() (pymisp.MISPTag method), 24  
 set\_not\_jsonable() (pymisp.MISPUser method), 25  
 set\_not\_jsonable() (pymisp.tools.ELFObject method), 29  
 set\_not\_jsonable() (pymisp.tools.ELFSectionObject method), 29  
 set\_not\_jsonable() (pymisp.tools.FileObject method), 28  
 set\_not\_jsonable() (pymisp.tools.MachOObject method), 32  
 set\_not\_jsonable() (pymisp.tools.MachOSectionObject method), 33  
 set\_not\_jsonable() (pymisp.tools.PEObject method), 30  
 set\_not\_jsonable() (pymisp.tools.PESectionObject method), 31  
 set\_not\_jsonable() (pymisp.tools.VTReportObject method), 34  
 set\_sightings() (pymisp.PyMISP method), 16  
 setdefault() (pymisp.MISPAttribute method), 20  
 setdefault() (pymisp.MISPEvent method), 19  
 setdefault() (pymisp.MISPObject method), 21  
 setdefault() (pymisp.MISPObjectAttribute method), 22  
 setdefault() (pymisp.MISPObjectReference method), 23

setdefault() (pymisp.MISPOrganisation method), 26  
 setdefault() (pymisp.MISPTag method), 24  
 setdefault() (pymisp.MISPUser method), 25  
 setdefault() (pymisp.tools.ELFObject method), 29  
 setdefault() (pymisp.tools.ELFSectionObject method), 29  
 setdefault() (pymisp.tools.FileObject method), 28  
 setdefault() (pymisp.tools.MachOObject method), 32  
 setdefault() (pymisp.tools.MachOSectionObject method), 33  
 setdefault() (pymisp.tools.PEObject method), 30  
 setdefault() (pymisp.tools.PESectionObject method), 31  
 setdefault() (pymisp.tools.VTReportObject method), 34  
 sighting() (pymisp.PyMISP method), 16  
 sighting\_per\_id() (pymisp.PyMISP method), 17  
 sighting\_per\_json() (pymisp.PyMISP method), 17  
 sighting\_per\_uuid() (pymisp.PyMISP method), 17

## T

tag() (pymisp.PyMISP method), 17  
 test\_connection() (pymisp.PyMISP method), 17  
 to\_dict() (pymisp.AbstractMISP method), 18  
 to\_dict() (pymisp.MISPObjectReference method), 23  
 to\_dict() (pymisp.MISPOrganisation method), 26  
 to\_dict() (pymisp.MISPTag method), 24  
 to\_dict() (pymisp.MISPUser method), 25  
 to\_json() (pymisp.AbstractMISP method), 18  
 to\_json() (pymisp.MISPAttribute method), 20  
 to\_json() (pymisp.MISPEvent method), 19  
 to\_json() (pymisp.MISPObjectAttribute method), 22  
 to\_json() (pymisp.MISPObjectReference method), 23  
 to\_json() (pymisp.MISPOrganisation method), 26  
 to\_json() (pymisp.MISPTag method), 24  
 to\_json() (pymisp.MISPUser method), 25

## U

unpublish() (pymisp.MISPEvent method), 19  
 untag() (pymisp.PyMISP method), 17  
 update() (pymisp.MISPAttribute method), 20  
 update() (pymisp.MISPEvent method), 19  
 update() (pymisp.MISPObject method), 21  
 update() (pymisp.MISPObjectAttribute method), 22  
 update() (pymisp.MISPObjectReference method), 23  
 update() (pymisp.MISPOrganisation method), 26  
 update() (pymisp.MISPTag method), 24  
 update() (pymisp.MISPUser method), 25  
 update() (pymisp.PyMISP method), 17  
 update() (pymisp.tools.ELFObject method), 29  
 update() (pymisp.tools.ELFSectionObject method), 29  
 update() (pymisp.tools.FileObject method), 28  
 update() (pymisp.tools.MachOObject method), 32  
 update() (pymisp.tools.MachOSectionObject method), 33  
 update() (pymisp.tools.PEObject method), 30  
 update() (pymisp.tools.PESectionObject method), 31  
 update() (pymisp.tools.VTReportObject method), 34

update\_event() (pymisp.PyMISP method), 17  
update\_not\_jsonable() (pymisp.AbstractMISP method),  
18  
update\_not\_jsonable() (pymisp.MISPAttribute method),  
20  
update\_not\_jsonable() (pymisp.MISPEvent method), 19  
update\_not\_jsonable() (pymisp.MISPObjct method), 21  
update\_not\_jsonable() (pymisp.MISPObjctAttribute  
method), 22  
update\_not\_jsonable() (pymisp.MISPObjctReference  
method), 23  
update\_not\_jsonable() (pymisp.MISPOrganisation  
method), 26  
update\_not\_jsonable() (pymisp.MISPTag method), 24  
update\_not\_jsonable() (pymisp.MISPUser method), 25  
update\_not\_jsonable() (pymisp.tools.ELFObject  
method), 29  
update\_not\_jsonable() (pymisp.tools.ELFSectionObject  
method), 30  
update\_not\_jsonable() (pymisp.tools.FileObject method),  
28  
update\_not\_jsonable() (pymisp.tools.MachOObject  
method), 32  
update\_not\_jsonable() (pymisp.tools.MachOSectionObject  
method), 33  
update\_not\_jsonable() (pymisp.tools.PEObject method),  
31  
update\_not\_jsonable() (pymisp.tools.PESectionObject  
method), 31  
update\_not\_jsonable() (pymisp.tools.VTReportObject  
method), 34  
upload\_sample() (pymisp.PyMISP method), 17  
upload\_samplelist() (pymisp.PyMISP method), 17

## V

values() (pymisp.MISPAttribute method), 20  
values() (pymisp.MISPEvent method), 19  
values() (pymisp.MISPObjct method), 21  
values() (pymisp.MISPObjctAttribute method), 22  
values() (pymisp.MISPObjctReference method), 23  
values() (pymisp.MISPOrganisation method), 26  
values() (pymisp.MISPTag method), 24  
values() (pymisp.MISPUser method), 25  
values() (pymisp.tools.ELFObject method), 29  
values() (pymisp.tools.ELFSectionObject method), 30  
values() (pymisp.tools.FileObject method), 28  
values() (pymisp.tools.MachOObject method), 32  
values() (pymisp.tools.MachOSectionObject method), 33  
values() (pymisp.tools.PEObject method), 31  
values() (pymisp.tools.PESectionObject method), 31  
values() (pymisp.tools.VTReportObject method), 34  
view\_feed() (pymisp.PyMISP method), 17  
view\_feeds() (pymisp.PyMISP method), 17  
VTReportObject (class in pymisp.tools), 33