
PyMISP Documentation

Release main

Raphaël Vinot

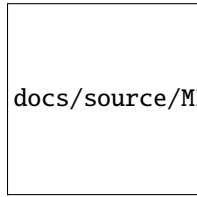
May 11, 2026

CONTENTS

1	PyMISP - Python Library to access MISP	3
1.1	Install from pip	3
1.2	Install the latest version from repo from development purposes	3
1.3	Samples and how to use PyMISP	4
1.4	Debugging	4
1.5	Test cases	5
1.6	Documentation	5
1.7	Everything is a Mutable Mapping	5
1.8	MISP Objects	5
1.9	Installing PyMISP on a machine with no internet access	5
2	License	7
3	pymisp - Classes	9
3.1	PyMISP	68
3.2	MISPAbstract	113
3.3	MISPEncode	114
3.4	MISPEvent	115
3.5	MISPEventBlocklist	119
3.6	MISPEventDelegation	120
3.7	MISPAttribute	122
3.8	MISPObject	124
3.9	MISPObjectAttribute	126
3.10	MISPObjectReference	128
3.11	MISPObjectTemplate	129
3.12	MISPTag	131
3.13	MISPUser	132
3.14	MISPUserSetting	133
3.15	MISPOrganisation	135
3.16	MISPOrganisationBlocklist	136
3.17	MISPFeed	137
3.18	MISPInbox	138
3.19	MISPLog	140
3.20	MISPNoticelist	141
3.21	MISPRole	142
3.22	MISPServer	144
3.23	MISPShadowAttribute	145
3.24	MISPSharingGroup	146
3.25	MISPSighting	147
3.26	MISPTaxonomy	149

3.27	MISPWarninglist	150
4	pymisp - Tools	153
4.1	Excel / CSV Importer	157
4.2	File Object	158
4.3	ELF Object	160
4.4	PE Object	165
4.5	Mach-O Object	170
4.6	VT Report Object	174
4.7	STIX	177
4.8	OpenIOC	177
5	Indices and tables	179
	Python Module Index	181
	Index	183

Contents:



docs/source/MISP-pymisp-hori-color.png

PYMISP - PYTHON LIBRARY TO ACCESS MISP

docs passing coverage 33% python 3.10+ pypi v2.5.34.1 downloads 994k/month

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.

1.1 Install from pip

It is strongly recommended to use a virtual environment

If you want to know more about virtual environments, [python has you covered](#)

Only basic dependencies:

```
pip3 install pymisp
```

And there are a few optional dependencies:

- fileobjects: to create PE/ELF/Mach-o objects
- openioc: to import files in OpenIOC format (not really maintained)
- virustotal: to query VirusTotal and generate the appropriate objects
- docs: to generate the documentation
- pdfexport: to generate PDF reports out of MISP events
- url: to generate URL objects out of URLs with Pyfaup
- email: to generate MISP Email objects
- brotli: to use the brotli compression when interacting with a MISP instance

Example:

```
pip3 install pymisp[virustotal,email]
```

1.2 Install the latest version from repo from development purposes

Note: poetry is required; e.g., “pip3 install poetry”

```
git clone https://github.com/MISP/PyMISP.git && cd PyMISP
git submodule update --init
poetry install -E fileobjects -E openioc -E virustotal -E docs -E pdfexport -E email
```

1.2.1 Running the tests

```
poetry run pytest --cov=pymisp tests/test_*.py
```

If you have a MISP instance to test against, you can also run the live ones:

Note: You need to update the key in `tests/testlive_comprehensive.py` to the automation key of your admin account.

```
poetry run pytest --cov=pymisp tests/testlive_comprehensive.py
```

1.3 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 events published in the last x amount of time (supported time indicators: days (d), hours (h) and minutes (m)). `last.py`

```
cd examples
python3 last.py -l 10h # 10 hours
python3 last.py -l 5d # 5 days
python3 last.py -l 45m # 45 minutes
```

1.4 Debugging

You have two options here:

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)
```

1.5 Test cases

1. The content of `misp_event.py` is tested on every commit
2. The test cases that require a running MISP instance can be run the following way:

```
# From poetry

pytest --cov=pymisp tests/test_*.py tests/testlive_comprehensive.py:TestComprehensive.
↪ [test_name]
```

1.6 Documentation

The documentation is available [here](#).

1.6.1 Jupyter notebook

A series of [Jupyter notebooks](#) for PyMISP tutorial are available in the repository.

1.7 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 inherits from `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 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.

1.8 MISP Objects

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

Your new `MISPObj` generator must 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.

1.9 Installing PyMISP on a machine with no internet access

This is done using poetry and you need to have this repository cloned on your machine. The commands below have to be run from inside the cloned directory.

1. From a machine with access to the internet, get the dependencies:

```
mkdir offline
poetry export --all-extras > offline/requirements.txt
poetry run pip download -r offline/requirements.txt -d offline/packages/
```

2. Prepare the PyMISP Package

```
poetry build
mv dist/*.whl offline/packages/
```

3. Copy the content of `offline/packages/` to the machine with no internet access.
4. Install the packages:

```
python -m pip install --no-index --no-deps packages/*.whl
```

LICENSE

PyMISP is distributed under an *open source license*. A simplified 2-BSD license.

PYMISP - CLASSES

`class pymisp.AbstractMISP(force_timestamps=False)`

property edited: `bool`

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.

Return type

`None`

from_json(*json_string*)

Load a JSON string

Return type

`None`

jsonable()

This method is used by the JSON encoder

Return type

`dict[str, Any]`

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

`None`

to_dict(*json_format=False*)

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.

Return type

`dict[str, Any]`

to_json(*sort_keys=False, indent=None*)

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

Return type

`str`

update_not_jsonable(*args)

Add entries to the __not_jsonable list

Return type

None

class pymisp.**Analysis**(*values)

class pymisp.**Distribution**(*values)

class pymisp.**ExpandedPyMISP**(*args, **kwargs)

exception pymisp.**InvalidMISPObject**(message)

Exception raised when an object doesn't respect the constraints in the definition

class pymisp.**MISPAttribute**(describe_types=None, strict=False)

add_galaxy(galaxy=None, **kwargs)

Add a galaxy to the Attribute, either by passing a MISPGalaxy or a dictionary

Return type

MISPGalaxy

add_proposal(shadow_attribute=None, **kwargs)

Alias for add_shadow_attribute

Return type

MISPShadowAttribute

add_shadow_attribute(shadow_attribute=None, **kwargs)

Add a shadow attribute to the attribute (by name or a MISPShadowAttribute object)

Return type

MISPShadowAttribute

add_sighting(sighting=None, **kwargs)

Add a sighting to the attribute (by name or a MISPSighting object)

Return type

MISPSighting

delete()

Mark the attribute as deleted (soft delete)

Return type

None

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.

Return type

None

property galaxies: **list**[*MISPGalaxy*]

Returns a list of galaxies associated to this Attribute

hash_values(*algorithm='sha512'*)

Compute the hash of every value for fast lookups

Return type

list[str]

property known_types: list[str]

Returns a list of all the known MISP attributes types

property malware_binary: BytesIO | None

Returns a BytesIO of the malware, if the attribute has one. Decrypts, unpacks and caches the binary on the first invocation, which may require some time for large attachments (~1s/MB).

property tags: list[MISPTag]

Returns a list of tags associated to this Attribute

to_dict(*json_format=False*)

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.

Return type

dict[str, Any]

class pymisp.MISPCorrelationExclusion(*force_timestamps=False*)

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.

Return type

None

class pymisp.MISPDecayingModel(*force_timestamps=False, **kwargs*)

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.

Return type

None

class pymisp.MISPEvent(*describe_types=None, strict_validation=False, force_timestamps=False, **kwargs*)

add_attribute(*type, value, **kwargs*)

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

Return type

MISPAtribute | list[MISPAtribute]

add_attribute_tag(*tag, attribute_identifier*)

Add a tag to an existing attribute. Raise an Exception if the attribute doesn't exist.

Parameters

- **tag** (*MISPTag* | str) – Tag name as a string, MISPTag instance, or dictionary
- **attribute_identifier** (str) – can be an ID, UUID, or the value.

Return type

list[MISPAttribute]

add_cryptographic_key(parent_type, key_data, type, uuid, fingerprint, timestamp, **kwargs)

Add a Cryptographic Key. parent_type, key_data, type, uuid, fingerprint, timestamp are required but you can pass all other parameters supported by MISPEventReport

Return type

MISPCryptographicKey

add_event_report(name, content, **kwargs)

Add an event report. name and value are required but you can pass all other parameters supported by MISPEventReport

Return type

MISPEventReport

add_galaxy(galaxy=None, **kwargs)

Add a galaxy and sub-clusters into an event, either by passing a MISPGalaxy or a dictionary. Supports all other parameters supported by MISPGalaxy

Return type

MISPGalaxy

add_object(obj=None, **kwargs)

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

Return type

MISPObject

add_proposal(shadow_attribute=None, **kwargs)

Alias for add_shadow_attribute

Return type

MISPShadowAttribute

add_shadow_attribute(shadow_attribute=None, **kwargs)

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

Return type

MISPShadowAttribute

delete_attribute(attribute_id)

Delete an attribute

Parameters

attribute_id (str) – ID or UUID

Return type

None

delete_object(object_id)

Delete an object

Parameters

object_id (str) – ID or UUID

Return type

None

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.

Return type

None

get_attribute_by_id(*attribute_id*)

Get an attribute by ID

Parameters

attribute_id (str | int) – The ID of the seeking attribute

Return type

MISPAttribute

get_attribute_by_uuid(*attribute_uuid*)

Get an attribute by UUID

Parameters

attribute_uuid (str) – The UUID of the seeking attribute

Return type

MISPAttribute

get_attribute_tag(*attribute_identifier*)

Return the tags associated to an attribute or an object attribute.

Parameters

attribute_identifier (str) – can be an ID, UUID, or the value.

Return type

list[*MISPTag*]

get_object_by_id(*object_id*)

Get an object by ID

Parameters

object_id (str | int) – the ID is the one set by the server when creating the new object

Return type

MISPObject

get_object_by_uuid(*object_uuid*)

Get an object by UUID

Parameters

object_uuid (str) – the UUID is set by the server when creating the new object

Return type

MISPObject

get_objects_by_name(*object_name*)

Get objects by name

Parameters

object_name (str) – name is set by the server when creating the new object

Return type

list[*MISPObject*]

load(*json_event*, *validate=False*, *metadata_only=False*)

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

Return type

None

load_file(*event_path*, *validate=False*, *metadata_only=False*)

Load a JSON dump from a file on the disk

Return type

None

publish()

Mark the attribute as published

Return type

None

set_date(*d=None*, *ignore_invalid=False*)

Set a date for the event

Parameters

- **d** (str | int | float | datetime | date | None) – String, datetime, or date object
- **ignore_invalid** (bool) – if True, assigns current date if d is not an expected type

Return type

None

property tags: **list**[*MISPTag*]

Returns a list of tags associated to this Event

to_dict(*json_format=False*)

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.

Return type

dict[str, Any]

to_feed(*valid_distributions=[0, 1, 2, 3, 4, 5]*, *with_meta=False*, *with_distribution=False*, *with_local_tags=True*, *with_event_reports=True*, *with_cryptographic_keys=True*)

Generate a json output for MISP Feed.

Parameters

- **valid_distributions** (list[int]) – only makes sense if the distribution key is set; i.e., the event is exported from a MISP instance.
- **with_distribution** (bool) – exports distribution and Sharing Group info; otherwise all SharingGroup information is discarded (protecting privacy)
- **with_local_tags** (bool) – tag export includes local exportable tags along with global exportable tags
- **with_event_reports** (bool) – include event reports in the returned MISP event
- **with_cryptographic_keys** (bool) – include the associated cryptographic keys in the returned protected MISP event

Return type

dict[str, Any]

unpublish()

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

Return type

None

class pymisp.MISPEventBlocklist(*force_timestamps=False, **kwargs*)

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.

Return type

None

class pymisp.MISPEventDelegation(*force_timestamps=False, **kwargs*)

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.

Return type

None

class pymisp.MISPEventReport(*force_timestamps=False, **kwargs*)

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.

Return type

None

class pymisp.MISPFeed(*force_timestamps=False*)

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.

Return type

None

class pymisp.MISPGalaxy

Galaxy class, used to view a galaxy and respective clusters

add_galaxy_cluster(kwargs)**

Add a MISP galaxy cluster into a MISPGalaxy. Supports all other parameters supported by MISPGalaxy-Cluster

Return type

MISPGalaxyCluster

from_dict(kwargs)**

Galaxy could be in one of the following formats: {'Galaxy': {}}, {'GalaxyCluster': []} {'Galaxy': {'Galaxy-Cluster': []}}

Return type

None

class pymisp.MISPGalaxyCluster

A MISP galaxy cluster, storing respective galaxy elements and relations. Used to view default galaxy clusters and add/edit/update/delete Galaxy 2.0 clusters

Creating a new galaxy cluster can take the following parameters

Parameters

- **value** (*str*) – The value of the galaxy cluster
- **description** (*str*) – The description of the galaxy cluster
- **distribution** (*int*) – The distribution type, one of 0, 1, 2, 3, 4
- **sharing_group_id** (*int*, *optional*) – The sharing group ID, if distribution is set to 4
- **authors** (*list[str]*, *optional*) – A list of authors of the galaxy cluster
- **cluster_elements** (*list[MISPGalaxyClusterElement]*, *optional*) – List of MISPGalaxyClusterElement
- **cluster_relations** (*list[MISPGalaxyClusterRelation]*, *optional*) – List of MISPGalaxyClusterRelation

add_cluster_element(*key*, *value*, ***kwargs*)

Add a cluster relation to a MISPGalaxyCluster, key and value are required

Parameters

- **key** (*str*) – The key name of the element
- **value** (*str*) – The value of the element

Return type

MISPGalaxyClusterElement

add_cluster_relation(*referenced_galaxy_cluster_uuid*, *referenced_galaxy_cluster_type*, *galaxy_cluster_uuid=None*, ***kwargs*)

Add a cluster relation to a MISPGalaxyCluster.

Parameters

- **referenced_galaxy_cluster_uuid** (*MISPGalaxyCluster* | *str* | *UUID*) – UUID of the related cluster
- **referenced_galaxy_cluster_type** (*str*) – Relation type
- **galaxy_cluster_uuid** (*str* | *None*) – UUID of this cluster, leave blank to use the stored UUID
- **galaxy_cluster_uuid** – uuid, Optional

Return type

MISPGalaxyClusterRelation

property elements_meta: *dict[str, Any]*

Function to return the galaxy cluster elements as a dictionary structure of lists that comes from a MISP-Galaxy within a MISPEvent. Lossy, you lose the element ID

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.

Return type

None

parse_meta_as_elements()

Function to parse the meta field into GalaxyClusterElements

Return type

None

class pymisp.MISPGalaxyClusterElement(*force_timestamps=False*)

A MISP Galaxy cluster element, providing further info on a cluster

Creating a new galaxy cluster element can take the following parameters

Parameters

- **key** (*str*) – The key/identifier of the element
- **value** (*str*) – The value of the element

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.

Return type

None

class pymisp.MISPGalaxyClusterRelation

A MISP Galaxy cluster relation, linking one cluster to another

Creating a new galaxy cluster can take the following parameters

Parameters

- **galaxy_cluster_uuid** – The UUID of the galaxy the relation links to
- **referenced_galaxy_cluster_type** – The relation type, e.g. dropped-by
- **referenced_galaxy_cluster_uuid** – The UUID of the related galaxy
- **distribution** – The distribution of the relation, one of 0, 1, 2, 3, 4, default 0
- **sharing_group_id** – The sharing group of the relation, only when distribution is 4

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.

Return type

None

property tags: list[MISPTag]

Returns a list of tags associated to this Attribute

class pymisp.MISPInbox(*force_timestamps=False, **kwargs*)

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.

Return type

None

class pymisp.MISPLog(force_timestamps=False, **kwargs)

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.

Return type

None

class pymisp.MISPNote(force_timestamps=False, **kwargs)

from_dict(contained=False, **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.

Return type

None

class pymisp.MISPNoticelist(force_timestamps=False)

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.

Return type

None

class pymisp.MISPObject(name, strict=False, standalone=True, default_attributes_parameters={}, force_timestamps=False, **kwargs)

add_attribute(object_relation, simple_value=None, **value)

Add an attribute. :type object_relation: str :param object_relation: The object relation of the attribute you're adding to the object :type simple_value: str | int | float | None :param simple_value: The value :type value: :param value: dictionary with all the keys supported by MISPAtribute

Return type

MISPAtribute | None

Note: as long as PyMISP knows about the object template, only the object_relation and the simple_value are required.

If PyMISP doesn't know the template, you also **must** pass a type. All the other options that can be passed along when creating an attribute (comment, IDS flag, ...) will be either taked out of the template, or out of the default setting for the type as defined on the MISP instance.

add_attributes(object_relation, *attributes)

Add multiple attributes with the same object_relation. Helper for object_relation when multiple is True in the template. It is the same as calling multiple times add_attribute with the same object_relation.

Return type`list[MISPAttribute | None]`**add_reference**(*referenced_uuid*, *relationship_type*, *comment=None*, ***kwargs*)

Add a link (uuid) to another object

Return type`MISPObjectReference`**delete**()

Mark the object as deleted (soft delete)

Return type`None`**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.

Return type`None`**get_attribute_by_id**(*attribute_id*)

Get an object attribute by ID

Parameters**attribute_id** (str | int) – The ID of the seeking object attribute**Return type**`MISPObjectAttribute`**get_attribute_by_uuid**(*attribute_uuid*)

Get an object attribute by UUID

Parameters**attribute_uuid** (str) – The UUID of the seeking object attribute**Return type**`MISPObjectAttribute`**get_attributes_by_relation**(*object_relation*)

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

Return type`list[MISPAttribute]`**has_attributes_by_relation**(*list_of_relations*)

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

Return type`bool`**to_dict**(*json_format=False*, *strict=False*)

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.

Return type`dict[str, Any]`

`to_json(sort_keys=False, indent=None, strict=False)`

Dump recursively any class of type MISPAbstract to a json string

Return type

str

`class pymisp.MISPObjectAttribute(definition, force_timestamps=False)`

`from_dict(object_relation="", value="", **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.

Return type

None

`exception pymisp.MISPObjectException(message)`

A base class for object specific exceptions

`class pymisp.MISPObjectReference`

`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.

Return type

None

`class pymisp.MISPObjectTemplate(force_timestamps=False)`

`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.

Return type

None

`class pymisp.MISPOpinion(force_timestamps=False, **kwargs)`

`from_dict(contained=False, **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.

Return type

None

`class pymisp.MISPOrganisation`

`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.

Return type

None

class pymisp.MISPOrganisationBlocklist(*force_timestamps=False, **kwargs*)

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.

Return type

None

class pymisp.MISPRelationship(*force_timestamps=False, **kwargs*)

from_dict(*contained=False, **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.

Return type

None

class pymisp.MISPRole(*force_timestamps=False, **kwargs*)

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.

Return type

None

class pymisp.MISPServer(*force_timestamps=False*)

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.

Return type

None

exception pymisp.MISPServerError(*message*)

class pymisp.MISPSHadowAttribute

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.

Return type

None

class pymisp.MISPSHaringGroup

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.

Return type

None

class pymisp.MISPSighting

from_dict(**kwargs)

Initialize the MISPSighting from a dictionary

Parameters

- **value** – Value of the attribute the sighting is related too. Pushing this object will update the sighting count of each attribute with this value on the instance.
- **uuid** – UUID of the attribute to update
- **id** – ID of the attribute to update
- **source** – Source of the sighting
- **type** – Type of the sighting
- **timestamp** – Timestamp associated to the sighting

Return type

None

class pymisp.MISPTag(force_timestamps=False, **kwargs)

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.

Return type

None

class pymisp.MISPTaxonomy(force_timestamps=False)

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.

Return type

None

class pymisp.MISPUser(force_timestamps=False, **kwargs)

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.

Return type

None

class pymisp.MISPUserSetting(force_timestamps=False, **kwargs)

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.

Return type

None

```
class pymisp.MISPWarninglist(force_timestamps=False)
```

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.

Return type

None

```
exception pymisp.NewAttributeError(message)
```

```
exception pymisp.NewEventError(message)
```

```
exception pymisp.NoKey(message)
```

```
exception pymisp.NoURL(message)
```

```
class pymisp.PyMISP(url, key, ssl=True, debug=False, proxies=None, cert=None, auth=None, tool="",
                   timeout=None, http_headers=None, https_adapter=None,
                   http_auth_header_name='Authorization')
```

Python API for MISP

Parameters

- **url** (str) – URL of the MISP instance you want to connect to
- **key** (str) – API key of the user you want to use
- **ssl** (bool | str) – can be True or False (to check or to not check the validity of the certificate. Or a CA_BUNDLE in case of self signed or other certificate (the concatenation of all the crt of the chain)
- **debug** (bool) – Write all the debug information to stderr
- **proxies** (MutableMapping[str, str] | None) – Proxy dict, as described here: <http://docs.python-requests.org/en/master/user/advanced/#proxies>
- **cert** (str | tuple[str, str] | None) – Client certificate, as described here: <http://docs.python-requests.org/en/master/user/advanced/#client-side-certificates>
- **auth** (AuthBase | None) – The auth parameter is passed directly to requests, as described here: <http://docs.python-requests.org/en/master/user/authentication/>
- **tool** (str) – The software using PyMISP (string), used to set a unique user-agent
- **http_headers** (dict[str, str] | None) – Arbitrary headers to pass to all the requests.
- **https_adapter** (BaseAdapter | None) – Arbitrary HTTPS adapter for the requests session.
- **http_auth_header_name** (str) – The name of the HTTP header to use for the API key. Can be either “Authorization” or “X-MISP-AUTH”.
- **timeout** (float | tuple[float, float] | None) – Timeout, as described here: <https://requests.readthedocs.io/en/master/user/advanced/#timeouts>

```
accept_attribute_proposal(proposal)
```

Accept a proposal. You cannot modify an existing proposal, only accept/discard

Parameters

proposal (*MISPShadowAttribute* | int | str | UUID) – attribute proposal to accept

Return type

dict[str, Any] | list[dict[str, Any]]

accept_event_delegation(*delegation*, *pythonify=False*)

Accept the delegation of an event

Parameters

- **delegation** (*MISPEventDelegation* | int | str) – event delegation to accept
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | list[dict[str, Any]]

accept_user_registration(*registration*, *organisation=None*, *role=None*, *perm_sync=False*, *perm_publish=False*, *perm_admin=False*, *unsafe_fallback=False*)

Accept a user registration

Parameters

- **registration** (*MISPInbox* | int | str | UUID) – the registration to accept
- **organisation** (*MISPOrganisation* | int | str | UUID | None) – user organization
- **role** (*MISPRole* | int | str | None) – user role
- **perm_sync** (bool) – indicator for sync
- **perm_publish** (bool) – indicator for publish
- **perm_admin** (bool) – indicator for admin
- **unsafe_fallback** (bool) – indicator for unsafe fallback

Return type

dict[str, Any] | list[dict[str, Any]]

add_analyst_data(*analyst_data*, *pythonify=False*)

Add an analyst data to an existing MISP element

Parameters

- **analyst_data** (*MISPNote* | *MISPOpinion* | *MISPRelationship*) – analyst_data to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPNote* | *MISPOpinion* | *MISPRelationship*

add_attribute(*event*, *attribute*, *pythonify=False*, *break_on_duplicate=True*)

Add an attribute to an existing MISP event: <https://www.misp-project.org/openapi/#tag/Attributes/operation/addAttribute>

Parameters

- **event** (*MISPEvent* | int | str | UUID) – event to extend
- **attribute** (*MISPAttribute* | Iterable[str]) – attribute or (MISP version 2.4.113+) list of attributes to add. If a list is passed, the pythonified response is a dict with the following structure: {'attributes': [MISPAttribute], 'errors': {errors by attributes}}
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output
- **break_on_duplicate** (bool) – if False, do not fail if the attribute already exists, updates existing attribute instead (timestamp will be always updated)

Return typedict[str, Any] | *MISPAttribute* | *MISPShadowAttribute***add_attribute_proposal**(*event*, *attribute*, *pythonify=False*)

Propose a new attribute in an event

Parameters

- **event** (*MISPEvent* | int | str | UUID) – event to receive new attribute
- **attribute** (*MISPAttribute*) – attribute to propose
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPShadowAttribute***add_correlation_exclusion**(*correlation_exclusion*, *pythonify=False*)

Add a new correlation exclusion

Parameters

- **correlation_exclusion** (*MISPCorrelationExclusion*) – correlation exclusion to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPCorrelationExclusion***add_event**(*event*, *pythonify=False*, *metadata=False*)Add a new event on a MISP instance: <https://www.misp-project.org/openapi/#tag/Events/operation/addEvent>**Parameters**

- **event** (*MISPEvent*) – event to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output
- **metadata** (bool) – Return just event metadata after successful creating

Return typedict[str, Any] | *MISPEvent***add_event_blocklist**(*uuids*, *comment=None*, *event_info=None*, *event_orgc=None*)

Add a new event in the blocklist

Parameters

- **uuids** (str | list[str]) – UUIDs
- **comment** (str | None) – comment
- **event_info** (str | None) – event information
- **event_orgc** (str | None) – event organization

Return type

dict[str, Any] | list[dict[str, Any]]

add_event_report(*event*, *event_report*, *pythonify=False*)

Add an event report to an existing MISP event

Parameters

- **event** (*MISPEvent* | int | str | UUID) – event to extend
- **event_report** (*MISPEventReport*) – event report to add.
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPEventReport*

add_feed(*feed*, *pythonify=False*)

Add a new feed on a MISP instance: <https://www.misp-project.org/openapi/#tag/Feeds/operation/addFeed>

Parameters

- **feed** (*MISPFeed*) – feed to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPFeed*

add_galaxy_cluster(*galaxy*, *galaxy_cluster*, *pythonify=False*)

Add a new galaxy cluster to a MISP Galaxy: <https://www.misp-project.org/openapi/#tag/Galaxy-Clusters/operation/addGalaxyCluster>

Parameters

- **galaxy** (*MISPGalaxy* | str | UUID) – A MISPGalaxy (or UUID) where you wish to add the galaxy cluster
- **galaxy_cluster** (*MISPGalaxyCluster*) – A MISPGalaxyCluster you wish to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPGalaxyCluster*

add_galaxy_cluster_relation(*galaxy_cluster_relation*)

Add a galaxy cluster relation, cluster relation must include cluster UUIDs in both directions

Parameters

galaxy_cluster_relation (*MISPGalaxyClusterRelation*) – The MISPGalaxyClusterRelation to add

Return type

dict[str, Any] | list[dict[str, Any]]

add_note(*note*, *pythonify=False*)

Add a note to an existing MISP element

Parameters

- **note** (*MISPNote*) – note to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPNote*

add_object(*event*, *misp_object*, *pythonify=False*, *break_on_duplicate=False*)

Add a MISP Object to an existing MISP event: <https://www.misp-project.org/openapi/#tag/Objects/operation/addObject>

Parameters

- **event** (*MISPEvent* | int | str | UUID) – event to extend
- **misp_object** (*MISPObject*) – object to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output
- **break_on_duplicate** (bool) – if True, check and reject if this object’s attributes match an existing object’s attributes; may require much time

Return typedict[str, Any] | *MISPObject***add_object_reference**(*misp_object_reference*, *pythonify=False*)

Add a reference to an object

Parameters

- **misp_object_reference** (*MISPObjectReference*) – object reference
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPObjectReference***add_opinion**(*opinion*, *pythonify=False*)

Add an opinion to an existing MISP element

Parameters

- **opinion** (*MISPOpinion*) – opinion to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPOpinion***add_org_to_sharing_group**(*sharing_group*, *organisation*, *extend=False*)Add an organisation to a sharing group: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/addOrganisationToSharingGroup>**Parameters**

- **sharing_group** (*MISPSharingGroup* | int | str | UUID) – Sharing group’s local instance ID, or Sharing group’s global UUID
- **organisation** (*MISPOrganisation* | int | str | UUID) – Organisation’s local instance ID, or Organisation’s global UUID, or Organisation’s name as known to the current instance
- **extend** (bool) – Allow the organisation to extend the group

Return type

dict[str, Any] | list[dict[str, Any]]

add_organisation(*organisation*, *pythonify=False*)Add an organisation: <https://www.misp-project.org/openapi/#tag/Organisations/operation/addOrganisation>**Parameters**

- **organisation** (*MISPOrganisation*) – organization to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPOrganisation*

add_organisation_blocklist(*uuids*, *comment=None*, *org_name=None*)

Add a new organisation in the blocklist

Parameters

- **uuids** (str | list[str]) – UUIDs
- **comment** (str | None) – comment
- **org_name** (str | None) – organization name

Return type

dict[str, Any] | list[dict[str, Any]]

add_relationship(*relationship*, *pythonify=False*)

Add a relationship to an existing MISP element

Parameters

- **relationship** (*MISPRelationship*) – relationship to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPRelationship*

add_role(*role*, *pythonify=False*)

Add a new role

Parameters

- **role** (*MISPRole*) – role to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPRole*

add_server(*server*, *pythonify=False*)

Add a server to synchronise with: <https://www.misp-project.org/openapi/#tag/Servers/operation/getServers> Note: You probably want to use `PyMISP.get_sync_config` and `PyMISP.import_server` instead

Parameters

- **server** (*MISPServer*) – sync server config
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPServer*

add_server_to_sharing_group(*sharing_group*, *server*, *all_orgs=False*)

Add a server to a sharing group: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/addServerToSharingGroup>

Parameters

- **sharing_group** (*MISPSharingGroup* | int | str | UUID) – Sharing group’s local instance ID, or Sharing group’s global UUID
- **server** (*MISPServer* | int | str | UUID) – Server’s local instance ID, or URL of the Server, or Server’s name as known to the current instance
- **all_orgs** (bool) – Add all the organisations of the server to the group

Return type

dict[str, Any] | list[dict[str, Any]]

add_sharing_group(*sharing_group*, *pythonify=False*)Add a new sharing group: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/addSharingGroup>**Parameters**

- **sharing_group** (*MISPSharingGroup*) – sharing group to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPSharingGroup***add_sighting**(*sighting*, *attribute=None*, *pythonify=False*)Add a new sighting (globally, or to a specific attribute): <https://www.misp-project.org/openapi/#tag/Sightings/operation/addSighting> and <https://www.misp-project.org/openapi/#tag/Sightings/operation/getSightingsByEventId>**Parameters**

- **sighting** (*MISPSighting* | dict[str, Any]) – sighting to add
- **attribute** (*MISPAttribute* | int | str | UUID | None) – specific attribute to modify with the sighting
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPSighting***add_tag**(*tag*, *pythonify=False*)Add a new tag on a MISP instance: <https://www.misp-project.org/openapi/#tag/Tags/operation/addTag>
The user calling this method needs the Tag Editor permission. It doesn't add a tag to an event, simply creates it on the MISP instance.**Parameters**

- **tag** (*MISPTag*) – tag to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPTag***add_user**(*user*, *pythonify=False*)Add a new user: <https://www.misp-project.org/openapi/#tag/Users/operation/addUser>**Parameters**

- **user** (*MISPUser*) – user to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPUser***attach_galaxy_cluster**(*misp_entity*, *galaxy_cluster*, *local=False*, *pythonify=False*)

Attach a galaxy cluster to an event or an attribute

Parameters

- **misp_entity** (*MISPEvent* | *MISPAttribute*) – a MISP Event or a MISP Attribute

- **galaxy_cluster** (*MISPGalaxyCluster* | int | str) – Galaxy cluster to attach
- **local** (bool) – whether the object should be attached locally or not to the target
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | list[dict[str, Any]]

attribute_exists(*attribute*)

Fast check if attribute exists.

Parameters

attribute (*MISPAttribute* | int | str | UUID) – Attribute to check

Return type

bool

attribute_proposals(*event=None, pythonify=False*)

Get all the attribute proposals

Parameters

- **event** (*MISPEvent* | int | str | UUID | None) – event
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[*MISPShadowAttribute*] | list[dict[str, Any]]

attributes(*pythonify=False*)

Get all the attributes from the MISP instance: <https://www.misp-project.org/openapi/#tag/Attributes/operation/getAttributes>

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[*MISPAttribute*] | list[dict[str, Any]]

attributes_statistics(*context='type', percentage=False*)

Get attribute statistics from the MISP instance

Parameters

- **context** (str) – “type” or “category”
- **percentage** (bool) – get percentages

Return type

dict[str, Any] | list[dict[str, Any]]

build_complex_query(*or_parameters=None, and_parameters=None, not_parameters=None*)

Build a complex search query. MISP expects a dictionary with AND, OR and NOT keys.

Return type

dict[str, list[TypeVar(SearchType, str, int)]]

cache_all_feeds()

Cache all the feeds: <https://www.misp-project.org/openapi/#tag/Feeds/operation/cacheFeeds>

Return type

dict[str, Any] | list[dict[str, Any]]

cache_feed(*feed*)Cache a specific feed by id: <https://www.misp-project.org/openapi/#tag/Feeds/operation/cacheFeeds>**Parameters****feed** (*MISPFeed* | int | str | UUID) – feed to cache**Return type**

dict[str, Any] | list[dict[str, Any]]

cache_freetext_feeds()

Cache all the freetext feeds

Return type

dict[str, Any] | list[dict[str, Any]]

cache_misp_feeds()

Cache all the MISP feeds

Return type

dict[str, Any] | list[dict[str, Any]]

change_sharing_group_on_entity(*misp_entity*, *sharing_group_id*, *pythonify=False*)

Change the sharing group of an event, an attribute, or an object

Parameters

- **misp_entity** (*MISPEvent* | *MISPAttribute* | *MISPObject*) – entity to change
- **sharing_group_id** (int) – group to change
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPEvent* | *MISPObject* | *MISPAttribute* | *MISPShadowAttribute***change_user_password**(*new_password*)

Change the password of the curent user:

Parameters**new_password** (str) – password to set**Return type**

dict[str, Any] | list[dict[str, Any]]

clean_correlation_exclusions()

Initiate correlation exclusions cleanup

Return type

dict[str, Any] | list[dict[str, Any]]

communities(*pythonify=False*)

Get all the communities

Parameters**pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM**Return type**

dict[str, Any] | list[MISPCommunity] | list[dict[str, Any]]

compare_feeds()

Generate the comparison matrix for all the MISP feeds

Return type

dict[str, Any] | list[dict[str, Any]]

contact_event_reporter(event, message)

Send a message to the reporter of an event

Parameters

- **event** (*MISPEvent* | int | str | UUID) – event with reporter to contact
- **message** (str) – message to send

Return type

dict[str, Any] | list[dict[str, Any]]

correlation_exclusions(pythonify=False)

Get all the correlation exclusions

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPCorrelationExclusion] | list[dict[str, Any]]

db_schema_diagnostic()

Get the schema diagnostic

Return type

dict[str, Any] | list[dict[str, Any]]

decaying_models(pythonify=False)

Get all the decaying models

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output

Return type

dict[str, Any] | list[MISPDecayingModel] | list[dict[str, Any]]

delegate_event(event=None, organisation=None, event_delegation=None, distribution=-1, message="", pythonify=False)

Delegate an event. Either event and organisation OR event_delegation are required

Parameters

- **event** (*MISPEvent* | int | str | UUID | None) – event to delegate
- **organisation** (*MISPOrganisation* | int | str | UUID | None) – organization
- **event_delegation** (*MISPEventDelegation* | None) – event delegation
- **distribution** (int) – distribution == -1 means recipient decides
- **message** (str) – message
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | MISPEventDelegation

delete_analyst_data(*analyst_data*)

Delete an analyst data from a MISP instance

Parameters

analyst_data (*MISPNote* | *MISPOpinion* | *MISPRelationship* | int | str | UUID) – analyst data to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_attribute(*attribute*, *hard=False*)

Delete an attribute from a MISP instance: <https://www.misp-project.org/openapi/#tag/Attributes/operation/deleteAttribute>

Parameters

- **attribute** (*MISPAttribute* | int | str | UUID) – attribute to delete
- **hard** (bool) – flag for hard delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_attribute_proposal(*attribute*)

Propose the deletion of an attribute

Parameters

attribute (*MISPAttribute* | int | str | UUID) – attribute to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_correlation_exclusion(*correlation_exclusion*)

Delete a correlation exclusion

Parameters

correlation_exclusion (*MISPCorrelationExclusion* | int | str | UUID) – The MISPCorrelationExclusion you wish to delete from MISP

Return type

dict[str, Any] | list[dict[str, Any]]

delete_event(*event*)

Delete an event from a MISP instance: <https://www.misp-project.org/openapi/#tag/Events/operation/deleteEvent>

Parameters

event (*MISPEvent* | int | str | UUID) – event to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_event_blocklist(*event_blocklist*)

Delete a blocklisted event by id

Parameters

event_blocklist (*MISPEventBlocklist* | str | UUID) – event block list to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_event_report(*event_report*, *hard=False*)

Delete an event report from a MISP instance

Parameters

- **event_report** (*MISPEventReport* | int | str | UUID) – event report to delete
- **hard** (bool) – flag for hard delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_feed(*feed*)

Delete a feed from a MISP instance

Parameters

feed (*MISPFeed* | int | str | UUID) – feed to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_galaxy_cluster(*galaxy_cluster*, *hard=False*)

Deletes a galaxy cluster from MISP: <https://www.misp-project.org/openapi/#tag/Galaxy-Clusters/operation/deleteGalaxyCluster>

Parameters

- **galaxy_cluster** (*MISPGalaxyCluster* | int | str | UUID) – The MISPGalaxyCluster you wish to delete from MISP
- **hard** (bool) – flag for hard delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_galaxy_cluster_relation(*galaxy_cluster_relation*)

Delete a galaxy cluster relation

Parameters

galaxy_cluster_relation (*MISPGalaxyClusterRelation* | int | str | UUID) – The MISPGalaxyClusterRelation to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_note(*note*)

Delete a note from a MISP instance

Parameters

note (*MISPNote* | int | str | UUID) – note delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_object(*misp_object*, *hard=False*)

Delete an object from a MISP instance: <https://www.misp-project.org/openapi/#tag/Objects/operation/deleteObject>

Parameters

- **misp_object** (*MISPObject* | int | str | UUID) – object to delete
- **hard** (bool) – flag for hard delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_object_reference(*object_reference*, *hard=False*)

Delete a reference to an object.

Return type

dict[str, Any] | list[dict[str, Any]]

delete_opinion(*opinion*)

Delete an opinion from a MISP instance

Parameters**opinion** (*MISPOpinion* | int | str | UUID) – opinion to delete**Return type**

dict[str, Any] | list[dict[str, Any]]

delete_organisation(*organisation*)Delete an organisation by id: <https://www.misp-project.org/openapi/#tag/Organisations/operation/deleteOrganisation>**Parameters****organisation** (*MISPOrganisation* | int | str | UUID) – organization to delete**Return type**

dict[str, Any] | list[dict[str, Any]]

delete_organisation_blocklist(*organisation_blocklist*)

Delete a blocklisted organisation by id

Parameters**organisation_blocklist** (*MISPOrganisationBlocklist* | str | UUID) – organization block list to delete**Return type**

dict[str, Any] | list[dict[str, Any]]

delete_relationship(*relationship*)

Delete a relationship from a MISP instance

Parameters**relationship** (*MISPRelationship* | int | str | UUID) – relationship to delete**Return type**

dict[str, Any] | list[dict[str, Any]]

delete_role(*role*)

Delete a role

Parameters**role** (*MISPRole* | int | str | UUID) – role to delete**Return type**

dict[str, Any] | list[dict[str, Any]]

delete_server(*server*)Delete a sync server: <https://www.misp-project.org/openapi/#tag/Servers/operation/getServers>**Parameters****server** (*MISPServer* | int | str | UUID) – sync server config

Return type

dict[str, Any] | list[dict[str, Any]]

delete_sharing_group(*sharing_group*)

Delete a sharing group: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/deleteSharingGroup>

Parameters

sharing_group (*MISPSharingGroup* | int | str | UUID) – sharing group to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_sighting(*sighting*)

Delete a sighting from a MISP instance: <https://www.misp-project.org/openapi/#tag/Sightings/operation/deleteSighting>

Parameters

sighting (*MISPSighting* | int | str | UUID) – sighting to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_tag(*tag*)

Delete a tag from a MISP instance: <https://www.misp-project.org/openapi/#tag/Tags/operation/deleteTag>

Parameters

tag (*MISPTag* | int | str | UUID) – tag to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_user(*user*)

Delete a user by id: <https://www.misp-project.org/openapi/#tag/Users/operation/deleteUser>

Parameters

user (*MISPUser* | int | str | UUID) – user to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_user_setting(*user_setting*, *user=None*)

Delete a user setting: <https://www.misp-project.org/openapi/#tag/UserSettings/operation/deleteUserSettingById>

Parameters

- **user_setting** (str) – name of user setting
- **user** (*MISPUser* | int | str | UUID | None) – user

Return type

dict[str, Any] | list[dict[str, Any]]

property describe_types_local: dict[str, Any] | list[dict[str, Any]]

Returns the content of describe types from the package

property describe_types_remote: dict[str, Any] | list[dict[str, Any]]

Returns the content of describe types from the remote instance

direct_call(*url*, *data=None*, *params={}*, *kw_params={}*)

Very lightweight call that posts a data blob (python dictionary or json string) on the URL

Parameters

- **url** (str) – URL to post to
- **data** (dict[str, Any] | None) – data to post
- **params** (Mapping[str, Any]) – dict with parameters for request
- **kw_params** (Mapping[str, Any]) – dict with keyword parameters for request

Return type

Any

disable_decaying_model(*decaying_model*)

Disable a decaying Model

Return type

dict[str, Any] | list[dict[str, Any]]

disable_feed(*feed*, *pythonify=False*)

Disable a feed: <https://www.misp-project.org/openapi/#tag/Feeds/operation/disableFeed>

Parameters

- **feed** (*MISPFeed* | int | str | UUID) – feed to disable
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPFeed*

disable_feed_cache(*feed*, *pythonify=False*)

Disable the caching of a feed

Parameters

- **feed** (*MISPFeed* | int | str | UUID) – feed to disable caching
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPFeed*

disable_noticelist(*noticelist*)

Disable a noticelist by id

Parameters

noticelist (*MISPNoticelist* | int | str | UUID) – Noticelist to disable

Return type

dict[str, Any] | list[dict[str, Any]]

disable_tag(*tag*, *pythonify=False*)

Disable a tag

Parameters

- **tag** (*MISPTag*) – tag to disable
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPTag*

disable_taxonomy(*taxonomy*)

Disable a taxonomy: <https://www.misp-project.org/openapi/#tag/Taxonomies/operation/disableTaxonomy>

Parameters

taxonomy (*MISPTaxonomy* | int | str | UUID) – taxonomy to disable

Return type

dict[str, Any] | list[dict[str, Any]]

disable_taxonomy_tags(*taxonomy*)

Disable all the tags of a taxonomy

Parameters

taxonomy (*MISPTaxonomy* | int | str | UUID) – taxonomy with tags to disable

Return type

dict[str, Any] | list[dict[str, Any]]

disable_warninglist(*warninglist*)

Disable a warninglist

Parameters

warninglist (*MISPWarninglist* | int | str | UUID) – warninglist to disable

Return type

dict[str, Any] | list[dict[str, Any]]

discard_attribute_proposal(*proposal*)

Discard a proposal. You cannot modify an existing proposal, only accept/discard

Parameters

proposal (*MISPShadowAttribute* | int | str | UUID) – attribute proposal to discard

Return type

dict[str, Any] | list[dict[str, Any]]

discard_event_delegation(*delegation*, *pythonify=False*)

Discard the delegation of an event

Parameters

- **delegation** (*MISPEventDelegation* | int | str) – event delegation to discard
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | list[dict[str, Any]]

discard_user_registration(*registration*)

Discard a user registration

Parameters

registration (*MISPInbox* | int | str | UUID) – the registration to discard

Return type

dict[str, Any] | list[dict[str, Any]]

enable_decaying_model(*decaying_model*)

Enable a decaying Model

Return type

dict[str, Any] | list[dict[str, Any]]

enable_feed(*feed*, *pythonify=False*)

Enable a feed; fetching it will create event(s): <https://www.misp-project.org/openapi/#tag/Feeds/operation/enableFeed>

Parameters

- **feed** (*MISPFeed* | int | str | UUID) – feed to enable
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPFeed*

enable_feed_cache(*feed*, *pythonify=False*)

Enable the caching of a feed

Parameters

- **feed** (*MISPFeed* | int | str | UUID) – feed to enable caching
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPFeed*

enable_noticelist(*noticelist*)

Enable a noticelist by id: <https://www.misp-project.org/openapi/#tag/Noticelists/operation/toggleEnableNoticelist>

Parameters

noticelist (*MISPNoticelist* | int | str | UUID) – Noticelist to enable

Return type

dict[str, Any] | list[dict[str, Any]]

enable_tag(*tag*, *pythonify=False*)

Enable a tag

Parameters

- **tag** (*MISPTag*) – tag to enable
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPTag*

enable_taxonomy(*taxonomy*)

Enable a taxonomy: <https://www.misp-project.org/openapi/#tag/Taxonomies/operation/enableTaxonomy>

Parameters

taxonomy (*MISPTaxonomy* | int | str | UUID) – taxonomy to enable

Return type

dict[str, Any] | list[dict[str, Any]]

enable_taxonomy_tags(*taxonomy*)

Enable all the tags of a taxonomy. NOTE: this is automatically done when you call enable_taxonomy

Parameters

taxonomy (*MISPTaxonomy* | int | str | UUID) – taxonomy with tags to enable

Return type

dict[str, Any] | list[dict[str, Any]]

enable_warninglist(*warninglist*)

Enable a warninglist

Parameters

warninglist (*MISPWarninglist* | int | str | UUID) – warninglist to enable

Return type

dict[str, Any] | list[dict[str, Any]]

enrich_attribute(*attribute*, *enrich_with*)

Enrich an attribute with data from one or more module.

Parameters

- **attribute** (*MISPAtribute* | int | str | UUID) – attribute to enrich
- **enrich_with** (str | list[str]) – module name or list of module names to use for enrichment

Return type

dict[str, Any]

enrich_event(*event*, *enrich_with*)

Enrich an event with data from one or more module.

Parameters

- **event** (*MISPEvent* | int | str | UUID) – event to enrich
- **enrich_with** (str | list[str]) – module name or list of module names to use for enrichment

Return type

dict[str, Any]

event_blocklists(*pythonify=False*)

Get all the blocklisted events

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPEventBlocklist] | list[dict[str, Any]]

event_delegations(*pythonify=False*)

Get all the event delegations

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPEventDelegation] | list[dict[str, Any]]

event_exists(*event*)

Fast check if event exists.

Parameters

event (*MISPEvent* | int | str | UUID) – Event to check

Return type

bool

events (*pythonify=False*)

Get all the events from the MISP instance: <https://www.misp-project.org/openapi/#tag/Events/operation/getEvents>

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPEvent] | list[dict[str, Any]]

feeds (*pythonify=False*)

Get the list of existing feeds: <https://www.misp-project.org/openapi/#tag/Feeds/operation/getFeeds>

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPFeed] | list[dict[str, Any]]

fetch_feed (*feed*)

Fetch one single feed by id: <https://www.misp-project.org/openapi/#tag/Feeds/operation/fetchFromFeed>

Parameters

feed (MISPFeed | int | str | UUID) – feed to fetch

Return type

dict[str, Any] | list[dict[str, Any]]

fork_galaxy_cluster (*galaxy, galaxy_cluster, pythonify=False*)

Forks an existing galaxy cluster, creating a new one with matching attributes

Parameters

- **galaxy** (MISPGalaxy | int | str | UUID) – The galaxy (or galaxy ID) where the cluster you want to fork resides
- **galaxy_cluster** (MISPGalaxyCluster) – The galaxy cluster you wish to fork
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | MISPGalaxyCluster

freetext (*event, string, adhereToWarninglists=False, distribution=None, returnMetaAttributes=False, pythonify=False, **kwargs*)

Pass a text to the freetext importer

Parameters

- **event** (MISPEvent | int | str | UUID) – event
- **string** (str) – query
- **adhereToWarninglists** (bool | str) – flag
- **distribution** (int | None) – distribution == -1 means recipient decides
- **returnMetaAttributes** (bool) – flag
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output
- **kwargs** – kwargs passed to prepare_request

Return type`dict[str, Any] | list[MISPAttribute] | list[dict[str, Any]]`**galaxies**(*withCluster=False, pythonify=False*)Get all the galaxies: <https://www.misp-project.org/openapi/#tag/Galaxies/operation/getGalaxies>**Parameters****pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM**Return type**`dict[str, Any] | list[MISPGalaxy] | list[dict[str, Any]]`**get_all_functions**(*not_implemented=False*)

Get all methods available via the API, including ones that are not implemented.

Return type`list[str]`**get_analyst_data**(*analyst_data, pythonify=False*)

Get an analyst data from a MISP instance

Parameters

- **analyst_data** (MISPAAnalystData | int | str | UUID) – analyst data to get
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type`dict[str, Any] | MISPSNote | MISPOpinion | MISPRelationship`**get_attribute**(*attribute, pythonify=False*)Get an attribute from a MISP instance: <https://www.misp-project.org/openapi/#tag/Attributes/operation/getAttributeById>**Parameters**

- **attribute** (MISPAttribute | int | str | UUID) – attribute to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type`dict[str, Any] | MISPAttribute`**get_attribute_proposal**(*proposal, pythonify=False*)

Get an attribute proposal

Parameters

- **proposal** (MISPShadowAttribute | int | str | UUID) – proposal to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type`dict[str, Any] | MISPShadowAttribute`**get_community**(*community, pythonify=False*)

Get a community by id from a MISP instance

Parameters

- **community** (MISPCommunity | int | str | UUID) – community to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | MISPCommunity

get_correlation_exclusion(*correlation_exclusion*, *pythonify=False*)

Get a correlation exclusion by ID

Parameters

- **correlation_exclusion** (*MISPCorrelationExclusion* | int | str | UUID) – Correlation exclusion to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPCorrelationExclusion***get_event**(*event*, *deleted=False*, *extended=False*, *pythonify=False*)Get an event from a MISP instance. Includes collections like Attribute, EventReport, Feed, Galaxy, Object, Tag, etc. so the response size may be large : <https://www.misp-project.org/openapi/#tag/Events/operation/getEventById>**Parameters**

- **event** (*MISPEvent* | int | str | UUID) – event to get
- **deleted** (bool | int | list[int]) – whether to include soft-deleted attributes
- **extended** (bool | int) – whether to get extended events
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return typedict[str, Any] | *MISPEvent***get_event_report**(*event_report*, *pythonify=False*)

Get an event report from a MISP instance

Parameters

- **event_report** (*MISPEventReport* | int | str | UUID) – event report to get
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return typedict[str, Any] | *MISPEventReport***get_event_reports**(*event_id*, *pythonify=False*)

Get event report from a MISP instance that are attached to an event ID

Parameters

- **event_id** (int | str) – event id to get the event reports for
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output.

Return typedict[str, Any] | list[*MISPEventReport*] | list[dict[str, Any]]**get_feed**(*feed*, *pythonify=False*)Get a feed by id: <https://www.misp-project.org/openapi/#tag/Feeds/operation/getFeedById>**Parameters**

- **feed** (*MISPFeed* | int | str | UUID) – feed to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPFeed*

get_galaxy(*galaxy*, *withCluster=False*, *pythonify=False*)

Get a galaxy by id: <https://www.misp-project.org/openapi/#tag/Galaxies/operation/getGalaxyById>

Parameters

- **galaxy** (*MISPGalaxy* | int | str | UUID) – galaxy to get
- **withCluster** (bool) – Include the clusters associated with the galaxy
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPGalaxy*

get_galaxy_cluster(*galaxy_cluster*, *pythonify=False*)

Gets a specific galaxy cluster

Parameters

- **galaxy_cluster** (*MISPGalaxyCluster* | int | str | UUID) – The MISPGalaxyCluster you want to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPGalaxyCluster*

get_new_authkey(*user='me'*)

Get a new authorization key for a specific user, defaults to user doing the call: <https://www.misp-project.org/openapi/#tag/AuthKeys/operation/addAuthKey>

Parameters

user (*MISPUser* | int | str | UUID) – The owner of the key

Return type

str

get_note(*note*, *pythonify=False*)

Get a note from a MISP instance

Parameters

- **note** (*MISPNote*) – note to get
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | *MISPNote*

get_noticelist(*noticelist*, *pythonify=False*)

Get a noticelist by id: <https://www.misp-project.org/openapi/#tag/Noticelists/operation/getNoticelistById>

Parameters

- **noticelist** – Noticelist to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPNoticelist***get_object**(*misp_object*, *pythonify=False*)Get an object from the remote MISP instance: <https://www.misp-project.org/openapi/#tag/Objects/operation/getObjectById>**Parameters**

- **misp_object** (*MISPObject* | int | str | UUID) – object to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPObject***get_object_template**(*object_template*, *pythonify=False*)

Gets the full object template

Parameters

- **object_template** (*MISPObjectTemplate* | int | str | UUID) – template or ID to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPObjectTemplate***get_opinion**(*opinion*, *pythonify=False*)

Get an opinion from a MISP instance

Parameters

- **opinion** (*MISPOpinion*) – opinion to get
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return typedict[str, Any] | *MISPOpinion***get_organisation**(*organisation*, *pythonify=False*)Get an organisation by id: <https://www.misp-project.org/openapi/#tag/Organisations/operation/getOrganisationById>**Parameters**

- **organisation** (*MISPOrganisation* | int | str | UUID) – organization to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPOrganisation***get_raw_object_template**(*uuid_or_name*)

Get a row template. It needs to be present on disk on the MISP instance you're connected to. The response of this method can be passed to MISPObject(<name>, misp_objects_template_custom=<response>)

Return type

dict[str, Any] | list[dict[str, Any]]

get_relationship(*relationship*, *pythonify=False*)

Get a relationship from a MISP instance

Parameters

- **relationship** (*MISPRelationship*) – relationship to get
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output.
Warning: it might use a lot of RAM

Return type

dict[str, Any] | *MISPRelationship*

get_server_setting(*setting*)

Get a setting from the MISP instance

Parameters

setting (str) – server setting name

Return type

dict[str, Any] | list[dict[str, Any]]

get_sharing_group(*sharing_group*, *pythonify=False*)

Get a sharing group: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/getSharingGroupById>

Parameters

- **sharing_group** (*MISPSharingGroup* | int | str | UUID) – sharing group to find
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPSharingGroup*

get_sync_config(*pythonify=False*)

Get the sync server config. WARNING: This method only works if the user calling it is a sync user

Parameters

pythonify (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPServer*

get_tag(*tag*, *pythonify=False*)

Get a tag by id: <https://www.misp-project.org/openapi/#tag/Tags/operation/getTagById>

Parameters

- **tag** (*MISPTag* | int | str | UUID) – tag to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPTag*

get_taxonomy(*taxonomy*, *pythonify=False*)

Get a taxonomy by id or namespace from a MISP instance: <https://www.misp-project.org/openapi/#tag/Taxonomies/operation/getTaxonomyById>

Parameters

- **taxonomy** (*MISPTaxonomy* | int | str | UUID) – taxonomy to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPTaxonomy*

get_user(*user='me', pythonify=False, expanded=False*)

Get a user by id: <https://www.misp-project.org/openapi/#tag/Users/operation/getUsers>

Parameters

- **user** (*MISPUser* | int | str | UUID) – user to get; *me* means the owner of the API key doing the query
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output
- **expanded** (bool) – Also returns a MISPRole and a MISPUserSetting. Only taken in account if pythonify is True.

Return type

dict[str, Any] | *MISPUser* | tuple[*MISPUser*, *MISPRole*, list[*MISPUserSetting*]]

get_user_setting(*user_setting, user=None, pythonify=False*)

Get a user setting: <https://www.misp-project.org/openapi/#tag/UserSettings/operation/getUserSettingById>

Parameters

- **user_setting** (str) – name of user setting
- **user** (*MISPUser* | int | str | UUID | None) – user
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPUserSetting*

get_warninglist(*warninglist, pythonify=False*)

Get a warninglist by id: <https://www.misp-project.org/openapi/#tag/Warninglists/operation/getWarninglistById>

Parameters

- **warninglist** (*MISPWarninglist* | int | str | UUID) – warninglist to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPWarninglist*

get_workers()

Get all the workers

Return type

dict[str, Any] | list[dict[str, Any]]

import_server(*server, pythonify=False*)

Import a sync server config received from get_sync_config

Parameters

- **server** (*MISPServer*) – sync server config
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPServer*

kill_all_workers()

Kill all the workers

Return type

`dict[str, Any] | list[dict[str, Any]]`

load_default_feeds()

Load all the default feeds.

Return type

`dict[str, Any] | list[dict[str, Any]]`

property misp_instance_version: `dict[str, Any] | list[dict[str, Any]]`

Returns the version of the instance.

property misp_instance_version_master: `dict[str, Any] | list[dict[str, Any]]`

Get the most recent version from github

noticelists(*pythonify=False*)

Get all the noticelists: <https://www.misp-project.org/openapi/#tag/Noticelists/operation/getNoticelists>

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

`dict[str, Any] | list[MISPNoticelist] | list[dict[str, Any]]`

object_exists(*misp_object*)

Fast check if object exists.

Parameters

misp_object (*MISPObject* | int | str | UUID) – Attribute to check

Return type

bool

object_templates(*pythonify=False*)

Get all the object templates

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

`dict[str, Any] | list[MISPObjectTemplate] | list[dict[str, Any]]`

organisation_blocklists(*pythonify=False*)

Get all the blocklisted organisations

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

`dict[str, Any] | list[MISPOrganisationBlocklist] | list[dict[str, Any]]`

organisation_exists(*organisation*)

Fast check if organisation exists.

Parameters

organisation (*MISPOrganisation* | int | str | UUID) – Organisation to check

Return type

bool

organisations(*scope='local', search=None, pythonify=False*)Get all the organisations: <https://www.misp-project.org/openapi/#tag/Organisations/operation/getOrganisations>**Parameters**

- **scope** (str) – scope of organizations to get
- **search** (str | None) – The search to make against the list of organisations
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output.
Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPOrganisation] | list[dict[str, Any]]

publish(*event, alert=False*)Publish the event with one single HTTP POST: <https://www.misp-project.org/openapi/#tag/Events/operation/publishEvent>**Parameters**

- **event** (MISPEvent | int | str | UUID) – event to publish
- **alert** (bool) – whether to send an email. The default is to not send a mail as it is assumed this method is called on update.

Return type

dict[str, Any] | list[dict[str, Any]]

publish_galaxy_cluster(*galaxy_cluster*)Publishes a galaxy cluster: <https://www.misp-project.org/openapi/#tag/Galaxy-Clusters/operation/publishGalaxyCluster>**Parameters****galaxy_cluster** (MISPGalaxyCluster | int | str | UUID) – The galaxy cluster you wish to publish**Return type**

dict[str, Any] | list[dict[str, Any]]

push_event_to_ZMQ(*event*)

Force push an event by id on ZMQ

Parameters**event** (MISPEvent | int | str | UUID) – the event to push**Return type**

dict[str, Any] | list[dict[str, Any]]

property pymisp_version_main: dict[str, Any] | list[dict[str, Any]]

Get the most recent version of PyMISP from github

property pymisp_version_master: dict[str, Any] | list[dict[str, Any]]

PyMISP version as defined in the main repository

property recommended_pymisp_version: dict[str, Any] | list[dict[str, Any]]

Returns the recommended API version from the server

remote_acl(*debug_type='findMissingFunctionNames'*)

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

Parameters

debug_type (str) – printAllFunctionNames, findMissingFunctionNames, or printRoleAccess

Return type

dict[str, Any] | list[dict[str, Any]]

remove_org_from_sharing_group(*sharing_group, organisation*)

Remove an organisation from a sharing group: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/removeOrganisationFromSharingGroup>

Parameters

- **sharing_group** (*MISPSharingGroup* | int | str | UUID) – Sharing group’s local instance ID, or Sharing group’s global UUID
- **organisation** (*MISPOrganisation* | int | str | UUID) – Organisation’s local instance ID, or Organisation’s global UUID, or Organisation’s name as known to the current instance

Return type

dict[str, Any] | list[dict[str, Any]]

remove_server_from_sharing_group(*sharing_group, server*)

Remove a server from a sharing group: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/removeServerFromSharingGroup>

Parameters

- **sharing_group** (*MISPSharingGroup* | int | str | UUID) – Sharing group’s local instance ID, or Sharing group’s global UUID
- **server** (*MISPServer* | int | str | UUID) – Server’s local instance ID, or URL of the Server, or Server’s name as known to the current instance

Return type

dict[str, Any] | list[dict[str, Any]]

request_community_access(*community, requestor_email_address=None, requestor_gpg_key=None, requestor_organisation_name=None, requestor_organisation_uuid=None, requestor_organisation_description=None, message=None, sync=False, anonymise_requestor_server=False, mock=False*)

Request the access to a community

Parameters

- **community** (*MISPCommunity* | int | str | UUID) – community to request access
- **requestor_email_address** (str | None) – requestor email
- **requestor_gpg_key** (str | None) – requestor key
- **requestor_organisation_name** (str | None) – requestor org name
- **requestor_organisation_uuid** (str | None) – requestor org ID
- **requestor_organisation_description** (str | None) – requestor org desc
- **message** (str | None) – requestor message
- **sync** (bool) – synchronize flag

- **anonymise_requestor_server** (bool) – anonymise flag
- **mock** (bool) – mock flag

Return type

dict[str, Any] | list[dict[str, Any]]

restart_dead_workers()

Restart the dead workers

Return type

dict[str, Any] | list[dict[str, Any]]

restart_workers()

Restart all the workers

Return type

dict[str, Any] | list[dict[str, Any]]

restore_attribute(attribute, pythonify=False)Restore a soft deleted attribute from a MISP instance: <https://www.misp-project.org/openapi/#tag/Attributes/operation/restoreAttribute>**Parameters****attribute** (*MISPAttribute* | int | str | UUID) – attribute to restore**Return type**dict[str, Any] | *MISPAttribute***roles(pythonify=False)**

Get the existing roles

Parameters**pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM**Return type**dict[str, Any] | list[*MISPRole*] | list[dict[str, Any]]

search(controller='events', return_format='json', limit=None, page=None, value=None, type_attribute=None, category=None, org=None, tags=None, event_tags=None, quick_filter=None, quickFilter=None, date_from=None, date_to=None, eventid=None, with_attachments=None, withAttachments=None, metadata=None, uuid=None, publish_timestamp=None, last=None, timestamp=None, published=None, enforce_warninglist=None, enforceWarninglist=None, to_ids=None, deleted=None, include_event_uuid=None, includeEventUuid=None, include_event_tags=None, includeEventTags=None, event_timestamp=None, sg_reference_only=None, eventinfo=None, searchall=None, requested_attributes=None, include_context=None, includeContext=None, headerless=None, include_sightings=None, includeSightings=None, include_correlations=None, includeCorrelations=None, include_decay_score=None, includeDecayScore=None, object_name=None, exclude_decayed=None, sharinggroup=None, pythonify=False, **kwargs)

Search in the MISP instance

Parameters

- **controller** (str) – Controller to search on, it can be *events*, *objects*, *attributes*. The response will either be a list of events, objects, or attributes. Reference documentation for each controller:
 - events: <https://www.misp-project.org/openapi/#tag/Events/operation/restSearchEvents>

- attributes: <https://www.misp-project.org/openapi/#tag/Attributes/operation/restSearchAttributes>
- objects: N/A
- **return_format** (str) – Set the return format of the search (Currently supported: json, xml, openioc, suricata, snort - more formats are being moved to restSearch with the goal being that all searches happen through this API). Can be passed as the first parameter after restSearch or via the JSON payload.
- **limit** (int | None) – Limit the number of results returned, depending on the scope (for example 10 attributes or 10 full events).
- **page** (int | None) – If a limit is set, sets the page to be returned. page 3, limit 100 will return records 201->300).
- **value** (TypeVar(SearchParameterTypes, str, List[str | int], Dict[str, str | int]) | None) – Search for the given value in the attributes' value field.
- **type_attribute** (TypeVar(SearchParameterTypes, str, List[str | int], Dict[str, str | int]) | None) – The attribute type, any valid MISP attribute type is accepted.
- **category** (TypeVar(SearchParameterTypes, str, List[str | int], Dict[str, str | int]) | None) – The attribute category, any valid MISP attribute category is accepted.
- **org** (TypeVar(SearchParameterTypes, str, List[str | int], Dict[str, str | int]) | None) – Search by the creator organisation by supplying the organisation identifier.
- **tags** (TypeVar(SearchParameterTypes, str, List[str | int], Dict[str, str | int]) | None) – Tags to search or to exclude. You can pass a list, or the output of *build_complex_query*
- **event_tags** (TypeVar(SearchParameterTypes, str, List[str | int], Dict[str, str | int]) | None) – Tags to search or to exclude at the event level. You can pass a list, or the output of *build_complex_query*
- **quick_filter** (str | None) – The string passed to this field will ignore all of the other arguments. MISP will return an xml / json (depending on the header sent) of all events that have a sub-string match on value in the event info, event orgc, or any of the attribute value1 / value2 fields, or in the attribute comment.
- **date_from** (datetime | date | int | str | float | None) – Events with the date set to a date after the one specified. This filter will use the date of the event.
- **date_to** (datetime | date | int | str | float | None) – Events with the date set to a date before the one specified. This filter will use the date of the event.
- **eventid** (TypeVar(SearchType, str, int) | None) – The events that should be included / excluded from the search
- **with_attachments** (bool | None) – If set, encodes the attachments / zipped malware samples as base64 in the data field within each attribute
- **metadata** (bool | None) – Only the metadata (event, tags, relations) is returned, attributes and proposals are omitted.
- **uuid** (str | None) – Restrict the results by uuid.
- **publish_timestamp** (None | datetime | date | int | str | float | tuple[datetime | date | int | str | float | None, datetime | date | int | str | float | None]) – Restrict the results by the last publish timestamp (newer than).

- **timestamp** (None | datetime | date | int | str | float | tuple[datetime | date | int | str | float | None, datetime | date | int | str | float | None]) – Restrict the results by the timestamp (last edit). Any event with a timestamp newer than the given timestamp will be returned. In case you are dealing with /attributes as scope, the attribute’s timestamp will be used for the lookup. The input can be a timestamp or a short-hand time description (7d or 24h for example). You can also pass a list with two values to set a time range (for example [“14d”, “7d”]).
- **published** (bool | None) – Set whether published or unpublished events should be returned. Do not set the parameter if you want both.
- **enforce_warninglist** (bool | None) – Remove any attributes from the result that would cause a hit on a warninglist entry.
- **to_ids** (TypeVar(ToIDSType, str, int, bool) | list[TypeVar(ToIDSType, str, int, bool)]) | None) – By default all attributes are returned that match the other filter parameters, regardless of their to_ids setting. To restrict the returned data set to to_ids only attributes set this parameter to 1. 0 for the ones with to_ids set to False.
- **deleted** (str | None) – If this parameter is set to 1, it will only return soft-deleted attributes. [“0”, “1”] will return the active ones as well as the soft-deleted ones.
- **include_event_uuid** (bool | None) – Instead of just including the event ID, also include the event UUID in each of the attributes.
- **include_event_tags** (bool | None) – Include the event level tags in each of the attributes.
- **event_timestamp** (datetime | date | int | str | float | None) – Only return attributes from events that have received a modification after the given timestamp.
- **sg_reference_only** (bool | None) – If this flag is set, sharing group objects will not be included, instead only the sharing group ID is set.
- **eventinfo** (str | None) – Filter on the event’s info field.
- **searchall** (bool | None) – Search for a full or a substring (delimited by % for substrings) in the event info, event tags, attribute tags, attribute values or attribute comment fields.
- **requested_attributes** (str | None) – [CSV only] Select the fields that you wish to include in the CSV export. By setting event level fields additionally, includeContext is not required to get event metadata.
- **include_context** (bool | None) – [Attribute only] Include the event data with each attribute. [CSV output] Add event level metadata in every line of the CSV.
- **headerless** (bool | None) – [CSV Only] The CSV created when this setting is set to true will not contain the header row.
- **include_sightings** (bool | None) – [JSON Only - Attribute] Include the sightings of the matching attributes.
- **include_decay_score** (bool | None) – Include the decay score at attribute level.
- **include_correlations** (bool | None) – [JSON Only - attribute] Include the correlations of the matching attributes.
- **object_name** (str | None) – [objects controller only] Search for objects with that name
- **exclude_decayed** (bool | None) – [attributes controller only] Exclude the decayed attributes from the response
- **sharinggroup** (int | list[int] | None) – Filter by sharing group ID(s)

- **pythonify** (bool | None) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | str | list[MISPEvent | MISPAtribute | MISPObject] | list[dict[str, Any]]

Deprecated:

Parameters

- **quickFilter** (str | None) – synonym for quick_filter
- **withAttachments** (bool | None) – synonym for with_attachments
- **last** (None | datetime | date | int | str | float | tuple[datetime | date | int | str | float | None, datetime | date | int | str | float | None]) – synonym for publish_timestamp
- **enforceWarninglist** (bool | None) – synonym for enforce_warninglist
- **includeEventUuid** (bool | None) – synonym for include_event_uuid
- **includeEventTags** (bool | None) – synonym for include_event_tags
- **includeContext** (bool | None) – synonym for include_context

search_feeds(value=None, pythonify=False)

Search in the feeds cached on the servers

Return type

dict[str, Any] | list[MISPFeed] | list[dict[str, Any]]

search_galaxy(value, withCluster=False, pythonify=False)

Text search to find a matching galaxy name, namespace, description, or uuid.

Return type

dict[str, Any] | list[MISPGalaxy] | list[dict[str, Any]]

search_galaxy_clusters(galaxy, context='all', searchall=None, pythonify=False)

Searches the galaxy clusters within a specific galaxy: <https://www.misp-project.org/openapi/#tag/Galaxy-Clusters/operation/getGalaxyClusters> and <https://www.misp-project.org/openapi/#tag/Galaxy-Clusters/operation/getGalaxyClusterById>

Parameters

- **galaxy** (MISPGalaxy | int | str | UUID) – The MISPGalaxy you wish to search in
- **context** (str) – The context of how you want to search within the **galaxy_**
- **searchall** (str | None) – The search you want to make against the galaxy and context
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | list[MISPGalaxyCluster] | list[dict[str, Any]]

search_index(all=None, attribute=None, email=None, published=None, hasproposal=None, eventid=None, tags=None, date_from=None, date_to=None, eventinfo=None, threatlevel=None, distribution=None, analysis=None, org=None, timestamp=None, publish_timestamp=None, sharinggroup=None, minimal=None, sort=None, desc=None, limit=None, page=None, pythonify=None)

Search event metadata shown on the event index page. Using ! in front of a value means NOT, except for parameters date_from, date_to and timestamp which cannot be negated. Criteria are AND-ed together;

values in lists are OR-ed together. Return matching events with metadata but no attributes or objects; also see `minimal` parameter.

Parameters

- **all** (str | None) – Search for a full or a substring (delimited by % for substrings) in the event info, event tags, attribute tags, attribute values or attribute comment fields.
- **attribute** (str | None) – Filter on attribute’s value.
- **email** (str | None) – Filter on user’s email.
- **published** (bool | None) – Set whether published or unpublished events should be returned. Do not set the parameter if you want both.
- **hasproposal** (bool | None) – Filter for events containing proposal(s).
- **eventid** (TypeVar(SearchType, str, int) | None) – The events that should be included / excluded from the search
- **tags** (TypeVar(SearchParameterTypes, str, List[str | int], Dict[str, str | int]) | None) – Tags to search or to exclude. You can pass a list, or the output of `build_complex_query`
- **date_from** (datetime | date | int | str | float | None) – Events with the date set to a date after the one specified. This filter will use the date of the event.
- **date_to** (datetime | date | int | str | float | None) – Events with the date set to a date before the one specified. This filter will use the date of the event.
- **eventinfo** (str | None) – Filter on the event’s info field.
- **threatlevel** (list[TypeVar(SearchType, str, int)] | None) – Threat level(s) (1,2,3,4) | list
- **distribution** (list[TypeVar(SearchType, str, int)] | None) – Distribution level(s) (0,1,2,3) | list
- **analysis** (list[TypeVar(SearchType, str, int)] | None) – Analysis level(s) (0,1,2) | list
- **org** (TypeVar(SearchParameterTypes, str, List[str | int], Dict[str, str | int]) | None) – Search by the creator organisation by supplying the organisation identifier.
- **timestamp** (None | datetime | date | int | str | float | tuple[datetime | date | int | str | float | None, datetime | date | int | str | float | None]) – Restrict the results by the timestamp (last edit). Any event with a timestamp newer than the given timestamp will be returned. In case you are dealing with /attributes as scope, the attribute’s timestamp will be used for the lookup.
- **publish_timestamp** (None | datetime | date | int | str | float | tuple[datetime | date | int | str | float | None, datetime | date | int | str | float | None]) – Filter on event’s publish timestamp.
- **sharinggroup** (list[TypeVar(SearchType, str, int)] | None) – Restrict by a sharing group | list
- **minimal** (bool | None) – Return only event ID, UUID, timestamp, sighting_timestamp and published.
- **sort** (str | None) – The field to sort the events by, such as ‘id’, ‘date’, ‘attribute_count’.
- **desc** (bool | None) – Whether to sort events ascending (default) or descending.
- **limit** (int | None) – Limit the number of events returned

- **page** (int | None) – If a limit is set, sets the page to be returned. page 3, limit 100 will return records 201->300).
- **pythonify** (bool | None) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPEvent] | list[dict[str, Any]]

search_logs(*limit=None, page=None, log_id=None, title=None, created=None, model=None, action=None, user_id=None, change=None, email=None, org=None, description=None, ip=None, pythonify=False*)

Search in logs

Note: to run substring queries simply append/prepend/encapsulate the search term with %

Parameters

- **limit** (int | None) – Limit the number of results returned, depending on the scope (for example 10 attributes or 10 full events).
- **page** (int | None) – If a limit is set, sets the page to be returned. page 3, limit 100 will return records 201->300).
- **log_id** (int | None) – Log ID
- **title** (str | None) – Log Title
- **created** (datetime | date | int | str | float | None) – Creation timestamp
- **model** (str | None) – Model name that generated the log entry
- **action** (str | None) – The thing that was done
- **user_id** (int | None) – ID of the user doing the action
- **change** (str | None) – Change that occurred
- **email** (str | None) – Email of the user
- **org** (str | None) – Organisation of the User doing the action
- **description** (str | None) – Description of the action
- **ip** (str | None) – Origination IP of the User doing the action
- **pythonify** (bool | None) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPLog] | list[dict[str, Any]]

search_sightings(*context=None, context_id=None, type_sighting=None, date_from=None, date_to=None, publish_timestamp=None, last=None, org=None, source=None, include_attribute=None, include_event_meta=None, pythonify=False*)

Search sightings

Parameters

- **context** (str | None) – The context of the search. Can be either “attribute”, “event”, or nothing (will then match on events and attributes).
- **context_id** (TypeVar(SearchType, str, int) | None) – Only relevant if context is either “attribute” or “event”. Then it is the relevant ID.
- **type_sighting** (str | None) – Type of sighting

- **date_from** (datetime | date | int | str | float | None) – Events with the date set to a date after the one specified. This filter will use the date of the event.
- **date_to** (datetime | date | int | str | float | None) – Events with the date set to a date before the one specified. This filter will use the date of the event.
- **publish_timestamp** (None | datetime | date | int | str | float | tuple(datetime | date | int | str | float | None, datetime | date | int | str | float | None)) – Restrict the results by the last publish timestamp (newer than).
- **org** (TypeVar(SearchType, str, int) | None) – Search by the creator organisation by supplying the organisation identifier.
- **source** (str | None) – Source of the sighting
- **include_attribute** (bool | None) – Include the attribute.
- **include_event_meta** (bool | None) – Include the meta information of the event.

Return typedict[str, Any] | list[dict[str, *MISPEvent* | *MISPAttribute* | *MISPSighting*]]

Deprecated:

Parameters**last** (None | datetime | date | int | str | float | tuple(datetime | date | int | str | float | None, datetime | date | int | str | float | None)) – synonym for publish_timestamp**Example**

```
>>> misp.search_sightings(publish_timestamp='30d') # search sightings for the
↳ last 30 days on the instance
[ ... ]
>>> misp.search_sightings(context='attribute', context_id=6, include_
↳ attribute=True) # return list of sighting for attribute 6 along with the
↳ attribute itself
[ ... ]
>>> misp.search_sightings(context='event', context_id=17, include_event_
↳ meta=True, org=2) # return list of sighting for event 17 filtered with org id
↳ 2
```

search_tags(tagname, strict_tagname=False, pythonify=False)Search for tags by name: <https://www.misp-project.org/openapi/#tag/Tags/operation/searchTag>**Parameters**

- **tag_name** – Name to search, use % for substrings matches.
- **strict_tagname** (bool) – only return tags matching exactly the tag name (so skipping synonyms and cluster's value)

Return typedict[str, Any] | list[*MISPTag*] | list[dict[str, Any]]**server_pull**(server, event=None)Initialize a pull from a sync server, optionally limited to one event: <https://www.misp-project.org/openapi/#tag/Servers/operation/pullServer>**Parameters**

- **server** (*MISPServer* | int | str | UUID) – sync server config
- **event** (*MISPEvent* | int | str | UUID | None) – event

Return type

dict[str, Any] | list[dict[str, Any]]

server_push(*server*, *event=None*)

Initialize a push to a sync server, optionally limited to one event: <https://www.misp-project.org/openapi/#tag/Servers/operation/pushServer>

Parameters

- **server** (*MISPServer* | int | str | UUID) – sync server config
- **event** (*MISPEvent* | int | str | UUID | None) – event

Return type

dict[str, Any] | list[dict[str, Any]]

server_settings()

Get all the settings from the server

Return type

dict[str, Any] | list[dict[str, Any]]

servers(*pythonify=False*)

Get the existing servers the MISP instance can synchronise with: <https://www.misp-project.org/openapi/#tag/Servers/operation/getServers>

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[*MISPServer*] | list[dict[str, Any]]

set_default_role(*role*)

Set a default role for the new user accounts

Parameters

role (*MISPRole* | int | str | UUID) – the default role to set

Return type

dict[str, Any] | list[dict[str, Any]]

set_server_setting(*setting*, *value*, *force=False*)

Set a setting on the MISP instance

Parameters

- **setting** (str) – server setting name
- **value** (str | int | bool) – value to set
- **force** (bool) – override value test

Return type

dict[str, Any] | list[dict[str, Any]]

set_user_setting(*user_setting*, *value*, *user=None*, *pythonify=False*)

Set a user setting: <https://www.misp-project.org/openapi/#tag/UserSettings/operation/setUserSetting>

Parameters

- **user_setting** (str) – name of user setting
- **value** (str | dict[str, Any]) – value to set

- **user** (*MISPUser* | int | str | UUID | None) – user
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPUserSetting***sharing_group_exists**(*sharing_group*)

Fast check if sharing group exists.

Parameters**sharing_group** (*MISPSharingGroup* | int | str | UUID) – Sharing group to check**Return type**

bool

sharing_groups(*pythonify=False*)Get the existing sharing groups: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/getSharingGroup>**Parameters****pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM**Return type**dict[str, Any] | list[*MISPSharingGroup*] | list[dict[str, Any]]**sightings**(*misp_entity=None, org=None, pythonify=False*)Get the list of sightings related to a MISPEvent or a MISPAtribute (depending on type of *misp_entity*): <https://www.misp-project.org/openapi/#tag/Sightings/operation/getSightingsByEventId>**Parameters**

- **misp_entity** (*AbstractMISP* | None) – MISP entity
- **org** (*MISPOrganisation* | int | str | UUID | None) – MISP organization
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return typedict[str, Any] | list[*MISPSighting*] | list[dict[str, Any]]**sign_blob**(*blob*)

Sign a blob

Parameters**blob** (str) – blob to sign**Return type**

str

start_worker(*worker_type*)

Start a worker :type worker_type: str :param worker_type: The type of worker, must be one of the following: “default” “email” “scheduler” “cache” “prio” “update”

Return type

dict[str, Any] | list[dict[str, Any]]

stop_worker_by_pid(*worker_pid*)Stop a worker by its PID (get the pid with `get_workers`): :type worker_pid: str | int :param worker_pid: The pid of the worker to stop

Return type

dict[str, Any] | list[dict[str, Any]]

tag(*misp_entity*, *tag*, *local=False*, *relationship_type=None*)

Tag an event or an attribute.

Parameters

- **misp_entity** (*AbstractMISP* | str | dict[str, Any]) – a MISPEvent, a MISP Attribute, or a UUID
- **tag** (*MISPTag* | str | dict[str, Any]) – tag to add
- **local** (bool) – whether to tag locally
- **relationship_type** (str | None) – Type of relationship between the tag and the attribute or event

Return type

dict[str, Any] | list[dict[str, Any]]

tags(*pythonify=False*, ***kw_params*)Get the list of existing tags: <https://www.misp-project.org/openapi/#tag/Tags/operation/getTags>**Parameters****pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM**Return type**

dict[str, Any] | list[MISPTag]

tags_statistics(*percentage=False*, *name_sort=False*)

Get tag statistics from the MISP instance

Parameters

- **percentage** (bool) – get percentages
- **name_sort** (bool) – sort by name

Return type

dict[str, Any] | list[dict[str, Any]]

taxonomies(*pythonify=False*)Get all the taxonomies: <https://www.misp-project.org/openapi/#tag/Taxonomies/operation/getTaxonomies>**Parameters****pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM**Return type**

dict[str, Any] | list[MISPTaxonomy] | list[dict[str, Any]]

test_server(*server*)

Test if a sync link is working as expected

Parameters**server** (*MISPServer* | int | str | UUID) – sync server config**Return type**

dict[str, Any] | list[dict[str, Any]]

toggle_global_pythonify()

Toggle the pythonify variable for the class

Return type

None

toggle_warninglist(*warninglist_id=None, warninglist_name=None, force_enable=None*)

Toggle (enable/disable) the status of a warninglist by id: <https://www.misp-project.org/openapi/#tag/Warninglists/operation/toggleEnableWarninglist>

Parameters

- **warninglist_id** (*str | int | list[int] | None*) – ID of the WarningList
- **warninglist_name** (*str | list[str] | None*) – name of the WarningList
- **force_enable** (*bool | None*) – Force the warning list in the enabled state (does nothing if already enabled) - None means toggle.

Return type

dict[str, Any] | list[dict[str, Any]]

unpublish(*event*)

Unpublish the event with one single HTTP POST: <https://www.misp-project.org/openapi/#tag/Events/operation/unpublishEvent>

Parameters

event (*MISPEvent | int | str | UUID*) – event to unpublish

Return type

dict[str, Any] | list[dict[str, Any]]

untag(*misp_entity, tag*)

Untag an event or an attribute

Parameters

- **misp_entity** (*AbstractMISP | str | dict[str, Any]*) – misp_entity can be a UUID
- **tag** (*MISPTag | str*) – tag to remove

Return type

dict[str, Any] | list[dict[str, Any]]

update_analyst_data(*analyst_data, analyst_data_id=None, pythonify=False*)

Update an analyst data on a MISP instance

Parameters

- **analyst_data** (*MISPNote | MISPOpinion | MISPRelationship*) – analyst data to update
- **analyst_data_id** (*int | None*) – analyst data ID to update
- **pythonify** (*bool*) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | MISPNote | MISPOpinion | MISPRelationship

update_attribute(*attribute, attribute_id=None, pythonify=False*)

Update an attribute on a MISP instance: <https://www.misp-project.org/openapi/#tag/Attributes/operation/editAttribute>

Parameters

- **attribute** (*MISPAttribute*) – attribute to update
- **attribute_id** (int | None) – attribute ID to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPAttribute* | *MISPShadowAttribute*

update_attribute_proposal(*initial_attribute*, *attribute*, *pythonify=False*)

Propose a change for an attribute

Parameters

- **initial_attribute** (*MISPAttribute* | int | str | UUID) – attribute to change
- **attribute** (*MISPAttribute*) – attribute to propose
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPShadowAttribute*

update_decaying_models()

Update all the Decaying models

Return type

dict[str, Any] | list[dict[str, Any]]

update_event(*event*, *event_id=None*, *pythonify=False*, *metadata=False*)

Update an event on a MISP instance: <https://www.misp-project.org/openapi/#tag/Events/operation/editEvent>

Parameters

- **event** (*MISPEvent*) – event to update
- **event_id** (int | None) – ID of event to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output
- **metadata** (bool) – Return just event metadata after successful update

Return type

dict[str, Any] | *MISPEvent*

update_event_blocklist(*event_blocklist*, *event_blocklist_id=None*, *pythonify=False*)

Update an event in the blocklist

Parameters

- **event_blocklist** (*MISPEventBlocklist*) – event block list
- **event_blocklist_id** (int | str | UUID | None) – event block list id
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPEventBlocklist*

update_event_report(*event_report*, *event_report_id=None*, *pythonify=False*)

Update an event report on a MISP instance

Parameters

- **event_report** (*MISPEventReport*) – event report to update

- **event_report_id** (int | None) – event report ID to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPEventReport***update_feed**(*feed*, *feed_id=None*, *pythonify=False*)

Update a feed on a MISP instance

Parameters

- **feed** (*MISPFeed*) – feed to update
- **feed_id** (int | None) – feed id
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPFeed***update_galaxies**()Update all the galaxies: <https://www.misp-project.org/openapi/#tag/Galaxies/operation/updateGalaxies>**Return type**

dict[str, Any] | list[dict[str, Any]]

update_galaxy_cluster(*galaxy_cluster*, *pythonify=False*)Update a custom galaxy cluster: <https://www.misp-project.org/openapi/#tag/Galaxy-Clusters/operation/editGalaxyCluster>

:param galaxy_cluster: The MISPGalaxyCluster you wish to update :type pythonify: bool :param pythonify: Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPGalaxyCluster***update_galaxy_cluster_relation**(*galaxy_cluster_relation*)

Update a galaxy cluster relation

Parameters**galaxy_cluster_relation** (*MISPGalaxyClusterRelation*) – The MISPGalaxyClusterRelation to update**Return type**

dict[str, Any] | list[dict[str, Any]]

update_misp()

Trigger a server update

Return type

dict[str, Any] | list[dict[str, Any]]

update_note(*note*, *note_id=None*, *pythonify=False*)

Update a note on a MISP instance

Parameters

- **note** (*MISPNote*) – note to update
- **note_id** (int | None) – note ID to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPNote*

update_noticelists()

Update all the noticelists: <https://www.misp-project.org/openapi/#tag/Noticelists/operation/updateNoticelists>

Return type

dict[str, Any] | list[dict[str, Any]]

update_object(misp_object, object_id=None, pythonify=False)

Update an object on a MISP instance

Parameters

- **misp_object** (*MISPObject*) – object to update
- **object_id** (int | None) – ID of object to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPObject*

update_object_templates()

Trigger an update of the object templates

Return type

dict[str, Any] | list[dict[str, Any]]

update_opinion(opinion, opinion_id=None, pythonify=False)

Update an opinion on a MISP instance

Parameters

- **opinion** (*MISPOpinion*) – opinion to update
- **opinion_id** (int | None) – opinion ID to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPOpinion*

update_organisation(organisation, organisation_id=None, pythonify=False)

Update an organisation: <https://www.misp-project.org/openapi/#tag/Organisations/operation/editOrganisation>

Parameters

- **organisation** (*MISPOrganisation*) – organization to update
- **organisation_id** (int | None) – id to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPOrganisation*

update_organisation_blocklist(organisation_blocklist, organisation_blocklist_id=None, pythonify=False)

Update an organisation in the blocklist

Parameters

- **organisation_blocklist** (*MISPOrganisationBlocklist*) – organization block list
- **organisation_blocklist_id** (int | str | UUID | None) – organization block list id
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPOrganisationBlocklist***update_relationship**(*relationship, relationship_id=None, pythonify=False*)

Update a relationship on a MISP instance

Parameters

- **relationship** (*MISPRelationship*) – relationship to update
- **relationship_id** (int | None) – relationship ID to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPRelationship***update_role**(*role, role_id=None, pythonify=False*)

Update a role on a MISP instance

Parameters

- **role** (*MISPRole*) – role to update
- **role_id** (int | None) – id to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPRole***update_server**(*server, server_id=None, pythonify=False*)Update a server to synchronise with: <https://www.misp-project.org/openapi/#tag/Servers/operation/getServers>**Parameters**

- **server** (*MISPServer*) – sync server config
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPServer***update_sharing_group**(*sharing_group, sharing_group_id=None, pythonify=False*)Update sharing group parameters: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/editSharingGroup>**Return type**dict[str, Any] | *MISPSharingGroup***Parameters****sharing_group** (*MISPSharingGroup* | dict[str, Any]) – MISP Sharing Group

:param sharing_group_id Sharing group ID :type pythonify: bool :param pythonify: Returns a PyMISP Object instead of the plain json output

update_tag(tag, tag_id=None, pythonify=False)

Edit only the provided parameters of a tag: <https://www.misp-project.org/openapi/#tag/Tags/operation/editTag>

Parameters

- **tag** (*MISPTag*) – tag to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Aram tag_id

tag ID to update

Return type

dict[str, Any] | *MISPTag*

update_taxonomies()

Update all the taxonomies: <https://www.misp-project.org/openapi/#tag/Taxonomies/operation/updateTaxonomies>

Return type

dict[str, Any] | list[dict[str, Any]]

update_user(user, user_id=None, pythonify=False)

Update a user on a MISP instance: <https://www.misp-project.org/openapi/#tag/Users/operation/editUser>

Parameters

- **user** (*MISPUser*) – user to update
- **user_id** (int | None) – id to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPUser*

update_warninglists()

Update all the warninglists: <https://www.misp-project.org/openapi/#tag/Warninglists/operation/updateWarninglists>

Return type

dict[str, Any] | list[dict[str, Any]]

upload_stix(path=None, data=None, version='2')

Upload a STIX file to MISP.

Parameters

- **path** (str | Path | BytesIO | StringIO | None) – Path to the STIX on the disk (can be a path-like object, or a pseudofile)
- **data** (str | bytes | None) – stix object
- **version** (str) – Can be 1 or 2

Return type

Response

user_registrations(pythonify=False)

Get all the user registrations

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPInbox] | list[dict[str, Any]]

user_settings(pythonify=False)

Get all the user settings: <https://www.misp-project.org/openapi/#tag/UserSettings/operation/getUserSettings>

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPUserSetting] | list[dict[str, Any]]

users(search=None, organisation=None, pythonify=False)

Get all the users, or a filtered set of users: <https://www.misp-project.org/openapi/#tag/Users/operation/getUsers>

Parameters

- **search** (str | None) – The search to make against the list of users
- **organisation** (int | None) – The ID of an organisation to filter against
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPUser] | list[dict[str, Any]]

users_heartbeat()

?

Return type

dict[str, str]

users_statistics(context='data')

Get user statistics from the MISP instance

Parameters

context (str) – one of 'data', 'orgs', 'users', 'tags', 'attributehistogram', 'sightings', 'galaxy-Matrix'

Return type

dict[str, Any] | list[dict[str, Any]]

values_in_warninglist(value)

Check if IOC values are in warninglist

Parameters

value (Iterable[str]) – iterator with values to check

Return type

dict[str, Any] | list[dict[str, Any]]

property version: dict[str, Any] | list[dict[str, Any]]

Returns the version of PyMISP you're currently using

warninglists(*pythonify=False*)

Get all the warninglists: <https://www.misp-project.org/openapi/#tag/Warninglists/operation/getWarninglists>

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPWarninglist]

exception pymisp.PyMISPError(*message*)

exception pymisp.PyMISPInvalidFormat(*message*)

class pymisp.ThreatLevel(**values*)

exception pymisp.UnknownMISPObjecTemplate(*message*)

Exception raised when the template is unknown

pymisp.register_user(*misp_url, email, organisation=None, org_id=None, org_name=None, message=None, custom_perms=None, perm_sync=False, perm_publish=False, perm_admin=False, verify=True*)

Ask for the creation of an account for the user with the given email address

Return type

dict[str, Any] | list[dict[str, Any]]

3.1 PyMISP

class pymisp.PyMISP(*url, key, ssl=True, debug=False, proxies=None, cert=None, auth=None, tool="", timeout=None, http_headers=None, https_adapter=None, http_auth_header_name='Authorization'*)

Python API for MISP

Parameters

- **url** (str) – URL of the MISP instance you want to connect to
- **key** (str) – API key of the user you want to use
- **ssl** (bool | str) – can be True or False (to check or to not check the validity of the certificate. Or a CA_BUNDLE in case of self signed or other certificate (the concatenation of all the crt of the chain)
- **debug** (bool) – Write all the debug information to stderr
- **proxies** (MutableMapping[str, str] | None) – Proxy dict, as described here: <http://docs.python-requests.org/en/master/user/advanced/#proxies>
- **cert** (str | tuple[str, str] | None) – Client certificate, as described here: <http://docs.python-requests.org/en/master/user/advanced/#client-side-certificates>
- **auth** (AuthBase | None) – The auth parameter is passed directly to requests, as described here: <http://docs.python-requests.org/en/master/user/authentication/>
- **tool** (str) – The software using PyMISP (string), used to set a unique user-agent
- **http_headers** (dict[str, str] | None) – Arbitrary headers to pass to all the requests.
- **https_adapter** (BaseAdapter | None) – Arbitrary HTTPS adapter for the requests session.

- **http_auth_header_name** (str) – The name of the HTTP header to use for the API key. Can be either “Authorization” or “X-MISP-AUTH”.
- **timeout** (float | tuple[float, float] | None) – Timeout, as described here: <https://requests.readthedocs.io/en/master/user/advanced/#timeouts>

accept_attribute_proposal(*proposal*)

Accept a proposal. You cannot modify an existing proposal, only accept/discard

Parameters

proposal (*MISPShadowAttribute* | int | str | UUID) – attribute proposal to accept

Return type

dict[str, Any] | list[dict[str, Any]]

accept_event_delegation(*delegation*, *pythonify=False*)

Accept the delegation of an event

Parameters

- **delegation** (*MISPEventDelegation* | int | str) – event delegation to accept
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | list[dict[str, Any]]

accept_user_registration(*registration*, *organisation=None*, *role=None*, *perm_sync=False*, *perm_publish=False*, *perm_admin=False*, *unsafe_fallback=False*)

Accept a user registration

Parameters

- **registration** (*MISPInbox* | int | str | UUID) – the registration to accept
- **organisation** (*MISPOrganisation* | int | str | UUID | None) – user organization
- **role** (*MISPRole* | int | str | None) – user role
- **perm_sync** (bool) – indicator for sync
- **perm_publish** (bool) – indicator for publish
- **perm_admin** (bool) – indicator for admin
- **unsafe_fallback** (bool) – indicator for unsafe fallback

Return type

dict[str, Any] | list[dict[str, Any]]

add_analyst_data(*analyst_data*, *pythonify=False*)

Add an analyst data to an existing MISP element

Parameters

- **analyst_data** (*MISPNote* | *MISPOpinion* | *MISPRelationship*) – analyst_data to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPNote* | *MISPOpinion* | *MISPRelationship*

add_attribute(*event*, *attribute*, *pythonify=False*, *break_on_duplicate=True*)

Add an attribute to an existing MISP event: <https://www.misp-project.org/openapi/#tag/Attributes/operation/addAttribute>

Parameters

- **event** (*MISPEvent* | int | str | UUID) – event to extend
- **attribute** (*MISPAttribute* | Iterable[str]) – attribute or (MISP version 2.4.113+) list of attributes to add. If a list is passed, the pythonified response is a dict with the following structure: {'attributes': [MISPAttribute], 'errors': {errors by attributes}}
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output
- **break_on_duplicate** (bool) – if False, do not fail if the attribute already exists, updates existing attribute instead (timestamp will be always updated)

Return type

dict[str, Any] | *MISPAttribute* | *MISPShadowAttribute*

add_attribute_proposal(*event*, *attribute*, *pythonify=False*)

Propose a new attribute in an event

Parameters

- **event** (*MISPEvent* | int | str | UUID) – event to receive new attribute
- **attribute** (*MISPAttribute*) – attribute to propose
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPShadowAttribute*

add_correlation_exclusion(*correlation_exclusion*, *pythonify=False*)

Add a new correlation exclusion

Parameters

- **correlation_exclusion** (*MISPCorrelationExclusion*) – correlation exclusion to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPCorrelationExclusion*

add_event(*event*, *pythonify=False*, *metadata=False*)

Add a new event on a MISP instance: <https://www.misp-project.org/openapi/#tag/Events/operation/addEvent>

Parameters

- **event** (*MISPEvent*) – event to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output
- **metadata** (bool) – Return just event metadata after successful creating

Return type

dict[str, Any] | *MISPEvent*

add_event_blocklist(*uuids*, *comment=None*, *event_info=None*, *event_orgc=None*)

Add a new event in the blocklist

Parameters

- **uuids** (str | list[str]) – UUIDs
- **comment** (str | None) – comment

- **event_info** (str | None) – event information
- **event_orgc** (str | None) – event organization

Return type

dict[str, Any] | list[dict[str, Any]]

add_event_report(*event*, *event_report*, *pythonify=False*)

Add an event report to an existing MISP event

Parameters

- **event** (*MISPEvent* | int | str | UUID) – event to extend
- **event_report** (*MISPEventReport*) – event report to add.
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPEventReport***add_feed**(*feed*, *pythonify=False*)Add a new feed on a MISP instance: <https://www.misp-project.org/openapi/#tag/Feeds/operation/addFeed>**Parameters**

- **feed** (*MISPFeed*) – feed to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPFeed***add_galaxy_cluster**(*galaxy*, *galaxy_cluster*, *pythonify=False*)Add a new galaxy cluster to a MISP Galaxy: <https://www.misp-project.org/openapi/#tag/Galaxy-Clusters/operation/addGalaxyCluster>**Parameters**

- **galaxy** (*MISPGalaxy* | str | UUID) – A MISPGalaxy (or UUID) where you wish to add the galaxy cluster
- **galaxy_cluster** (*MISPGalaxyCluster*) – A MISPGalaxyCluster you wish to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPGalaxyCluster***add_galaxy_cluster_relation**(*galaxy_cluster_relation*)

Add a galaxy cluster relation, cluster relation must include cluster UUIDs in both directions

Parameters**galaxy_cluster_relation** (*MISPGalaxyClusterRelation*) – The MISPGalaxyClusterRelation to add**Return type**

dict[str, Any] | list[dict[str, Any]]

add_note(*note*, *pythonify=False*)

Add a note to an existing MISP element

Parameters

- **note** (*MISPNote*) – note to add

- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPNote*

add_object(*event*, *misp_object*, *pythonify=False*, *break_on_duplicate=False*)

Add a MISP Object to an existing MISP event: <https://www.misp-project.org/openapi/#tag/Objects/operation/addObject>

Parameters

- **event** (*MISPEvent* | int | str | UUID) – event to extend
- **misp_object** (*MISPObject*) – object to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output
- **break_on_duplicate** (bool) – if True, check and reject if this object’s attributes match an existing object’s attributes; may require much time

Return type

dict[str, Any] | *MISPObject*

add_object_reference(*misp_object_reference*, *pythonify=False*)

Add a reference to an object

Parameters

- **misp_object_reference** (*MISPObjectReference*) – object reference
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPObjectReference*

add_opinion(*opinion*, *pythonify=False*)

Add an opinion to an existing MISP element

Parameters

- **opinion** (*MISPOpinion*) – opinion to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPOpinion*

add_org_to_sharing_group(*sharing_group*, *organisation*, *extend=False*)

Add an organisation to a sharing group: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/addOrganisationToSharingGroup>

Parameters

- **sharing_group** (*MISPSharingGroup* | int | str | UUID) – Sharing group’s local instance ID, or Sharing group’s global UUID
- **organisation** (*MISPOrganisation* | int | str | UUID) – Organisation’s local instance ID, or Organisation’s global UUID, or Organisation’s name as known to the current instance
- **extend** (bool) – Allow the organisation to extend the group

Return type

dict[str, Any] | list[dict[str, Any]]

add_organisation(*organisation*, *pythonify=False*)

Add an organisation: <https://www.misp-project.org/openapi/#tag/Organisations/operation/addOrganisation>

Parameters

- **organisation** (*MISPOrganisation*) – organization to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPOrganisation*

add_organisation_blocklist(*uuids*, *comment=None*, *org_name=None*)

Add a new organisation in the blocklist

Parameters

- **uuids** (str | list[str]) – UUIDs
- **comment** (str | None) – comment
- **org_name** (str | None) – organization name

Return type

dict[str, Any] | list[dict[str, Any]]

add_relationship(*relationship*, *pythonify=False*)

Add a relationship to an existing MISP element

Parameters

- **relationship** (*MISPRelationship*) – relationship to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPRelationship*

add_role(*role*, *pythonify=False*)

Add a new role

Parameters

- **role** (*MISPRole*) – role to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPRole*

add_server(*server*, *pythonify=False*)

Add a server to synchronise with: <https://www.misp-project.org/openapi/#tag/Servers/operation/getServers> Note: You probably want to use `PyMISP.get_sync_config` and `PyMISP.import_server` instead

Parameters

- **server** (*MISPServer*) – sync server config
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPServer*

add_server_to_sharing_group(*sharing_group*, *server*, *all_orgs=False*)

Add a server to a sharing group: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/addServerToSharingGroup>

Parameters

- **sharing_group** (*MISPSharingGroup* | int | str | UUID) – Sharing group’s local instance ID, or Sharing group’s global UUID
- **server** (*MISPServer* | int | str | UUID) – Server’s local instance ID, or URL of the Server, or Server’s name as known to the current instance
- **all_orgs** (bool) – Add all the organisations of the server to the group

Return type

dict[str, Any] | list[dict[str, Any]]

add_sharing_group(*sharing_group*, *pythonify=False*)

Add a new sharing group: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/addSharingGroup>

Parameters

- **sharing_group** (*MISPSharingGroup*) – sharing group to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPSharingGroup*

add_sighting(*sighting*, *attribute=None*, *pythonify=False*)

Add a new sighting (globally, or to a specific attribute): <https://www.misp-project.org/openapi/#tag/Sightings/operation/addSighting> and <https://www.misp-project.org/openapi/#tag/Sightings/operation/getSightingsByEventId>

Parameters

- **sighting** (*MISPSighting* | dict[str, Any]) – sighting to add
- **attribute** (*MISPAttribute* | int | str | UUID | None) – specific attribute to modify with the sighting
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPSighting*

add_tag(*tag*, *pythonify=False*)

Add a new tag on a MISP instance: <https://www.misp-project.org/openapi/#tag/Tags/operation/addTag>
The user calling this method needs the Tag Editor permission. It doesn’t add a tag to an event, simply creates it on the MISP instance.

Parameters

- **tag** (*MISPTag*) – tag to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPTag*

add_user(*user*, *pythonify=False*)

Add a new user: <https://www.misp-project.org/openapi/#tag/Users/operation/addUser>

Parameters

- **user** (*MISPUser*) – user to add
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPUser***attach_galaxy_cluster**(*misp_entity*, *galaxy_cluster*, *local=False*, *pythonify=False*)

Attach a galaxy cluster to an event or an attribute

Parameters

- **misp_entity** (*MISPEvent* | *MISPAttribute*) – a MISP Event or a MISP Attribute
- **galaxy_cluster** (*MISPGalaxyCluster* | int | str) – Galaxy cluster to attach
- **local** (bool) – whether the object should be attached locally or not to the target
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | list[dict[str, Any]]

attribute_exists(*attribute*)

Fast check if attribute exists.

Parameters**attribute** (*MISPAttribute* | int | str | UUID) – Attribute to check**Return type**

bool

attribute_proposals(*event=None*, *pythonify=False*)

Get all the attribute proposals

Parameters

- **event** (*MISPEvent* | int | str | UUID | None) – event
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return typedict[str, Any] | list[*MISPShadowAttribute*] | list[dict[str, Any]]**attributes**(*pythonify=False*)Get all the attributes from the MISP instance: <https://www.misp-project.org/openapi/#tag/Attributes/operation/getAttributes>**Parameters****pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM**Return type**dict[str, Any] | list[*MISPAttribute*] | list[dict[str, Any]]**attributes_statistics**(*context='type'*, *percentage=False*)

Get attribute statistics from the MISP instance

Parameters

- **context** (str) – “type” or “category”

- **percentage** (bool) – get percentages

Return type

dict[str, Any] | list[dict[str, Any]]

build_complex_query(*or_parameters=None, and_parameters=None, not_parameters=None*)

Build a complex search query. MISP expects a dictionary with AND, OR and NOT keys.

Return type

dict[str, list[TypeVar(SearchType, str, int)]]

cache_all_feeds()

Cache all the feeds: <https://www.misp-project.org/openapi/#tag/Feeds/operation/cacheFeeds>

Return type

dict[str, Any] | list[dict[str, Any]]

cache_feed(*feed*)

Cache a specific feed by id: <https://www.misp-project.org/openapi/#tag/Feeds/operation/cacheFeeds>

Parameters

feed (*MISPFeed* | int | str | UUID) – feed to cache

Return type

dict[str, Any] | list[dict[str, Any]]

cache_freetext_feeds()

Cache all the freetext feeds

Return type

dict[str, Any] | list[dict[str, Any]]

cache_misp_feeds()

Cache all the MISP feeds

Return type

dict[str, Any] | list[dict[str, Any]]

change_sharing_group_on_entity(*misp_entity, sharing_group_id, pythonify=False*)

Change the sharing group of an event, an attribute, or an object

Parameters

- **misp_entity** (*MISPEvent* | *MISPAttribute* | *MISPObject*) – entity to change
- **sharing_group_id** (int) – group to change
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPEvent* | *MISPObject* | *MISPAttribute* | *MISPShadowAttribute*

change_user_password(*new_password*)

Change the password of the current user:

Parameters

new_password (str) – password to set

Return type

dict[str, Any] | list[dict[str, Any]]

clean_correlation_exclusions()

Initiate correlation exclusions cleanup

Return type

dict[str, Any] | list[dict[str, Any]]

communities(*pythonify=False*)

Get all the communities

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPCommunity] | list[dict[str, Any]]

compare_feeds()

Generate the comparison matrix for all the MISP feeds

Return type

dict[str, Any] | list[dict[str, Any]]

contact_event_reporter(*event, message*)

Send a message to the reporter of an event

Parameters

- **event** (*MISPEvent* | int | str | UUID) – event with reporter to contact
- **message** (str) – message to send

Return type

dict[str, Any] | list[dict[str, Any]]

correlation_exclusions(*pythonify=False*)

Get all the correlation exclusions

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPCorrelationExclusion] | list[dict[str, Any]]

db_schema_diagnostic()

Get the schema diagnostic

Return type

dict[str, Any] | list[dict[str, Any]]

decaying_models(*pythonify=False*)

Get all the decaying models

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output

Return type

dict[str, Any] | list[MISPDecayingModel] | list[dict[str, Any]]

delegate_event(*event=None, organisation=None, event_delegation=None, distribution=-1, message="", pythonify=False*)

Delegate an event. Either event and organisation OR event_delegation are required

Parameters

- **event** (*MISPEvent* | int | str | UUID | None) – event to delegate
- **organisation** (*MISPOrganisation* | int | str | UUID | None) – organization
- **event_delegation** (*MISPEventDelegation* | None) – event delegation
- **distribution** (int) – distribution == -1 means recipient decides
- **message** (str) – message
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPEventDelegation*

delete_analyst_data(*analyst_data*)

Delete an analyst data from a MISP instance

Parameters

analyst_data (*MISPNote* | *MISPOpinion* | *MISPRelationship* | int | str | UUID) – analyst data to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_attribute(*attribute*, *hard=False*)

Delete an attribute from a MISP instance: <https://www.misp-project.org/openapi/#tag/Attributes/operation/deleteAttribute>

Parameters

- **attribute** (*MISPAttribute* | int | str | UUID) – attribute to delete
- **hard** (bool) – flag for hard delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_attribute_proposal(*attribute*)

Propose the deletion of an attribute

Parameters

attribute (*MISPAttribute* | int | str | UUID) – attribute to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_correlation_exclusion(*correlation_exclusion*)

Delete a correlation exclusion

Parameters

correlation_exclusion (*MISPCorrelationExclusion* | int | str | UUID) – The MISPCorrelationExclusion you wish to delete from MISP

Return type

dict[str, Any] | list[dict[str, Any]]

delete_event(*event*)

Delete an event from a MISP instance: <https://www.misp-project.org/openapi/#tag/Events/operation/deleteEvent>

Parameters

event (*MISPEvent* | int | str | UUID) – event to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_event_blocklist(*event_blocklist*)

Delete a blocklisted event by id

Parameters

event_blocklist (*MISPEventBlocklist* | str | UUID) – event block list to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_event_report(*event_report*, *hard=False*)

Delete an event report from a MISP instance

Parameters

- **event_report** (*MISPEventReport* | int | str | UUID) – event report to delete
- **hard** (bool) – flag for hard delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_feed(*feed*)

Delete a feed from a MISP instance

Parameters

feed (*MISPFee*d | int | str | UUID) – feed to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_galaxy_cluster(*galaxy_cluster*, *hard=False*)

Deletes a galaxy cluster from MISP: <https://www.misp-project.org/openapi/#tag/Galaxy-Clusters/operation/deleteGalaxyCluster>

Parameters

- **galaxy_cluster** (*MISPGalaxyCluster* | int | str | UUID) – The MISPGalaxyCluster you wish to delete from MISP
- **hard** (bool) – flag for hard delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_galaxy_cluster_relation(*galaxy_cluster_relation*)

Delete a galaxy cluster relation

Parameters

galaxy_cluster_relation (*MISPGalaxyClusterRelation* | int | str | UUID) – The MISPGalaxyClusterRelation to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_note(*note*)

Delete a note from a MISP instance

Parameters

note (*MISPNote* | int | str | UUID) – note delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_object(*misp_object*, *hard=False*)

Delete an object from a MISP instance: <https://www.misp-project.org/openapi/#tag/Objects/operation/deleteObject>

Parameters

- **misp_object** (*MISPObject* | int | str | UUID) – object to delete
- **hard** (bool) – flag for hard delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_object_reference(*object_reference*, *hard=False*)

Delete a reference to an object.

Return type

dict[str, Any] | list[dict[str, Any]]

delete_opinion(*opinion*)

Delete an opinion from a MISP instance

Parameters

opinion (*MISPOpinion* | int | str | UUID) – opinion to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_organisation(*organisation*)

Delete an organisation by id: <https://www.misp-project.org/openapi/#tag/Organisations/operation/deleteOrganisation>

Parameters

organisation (*MISPOrganisation* | int | str | UUID) – organization to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_organisation_blocklist(*organisation_blocklist*)

Delete a blocklisted organisation by id

Parameters

organisation_blocklist (*MISPOrganisationBlocklist* | str | UUID) – organization block list to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_relationship(*relationship*)

Delete a relationship from a MISP instance

Parameters

relationship (*MISPRelationship* | int | str | UUID) – relationship to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_role(*role*)

Delete a role

Parameters

role (*MISPRole* | int | str | UUID) – role to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_server(*server*)

Delete a sync server: <https://www.misp-project.org/openapi/#tag/Servers/operation/getServers>

Parameters

server (*MISPServer* | int | str | UUID) – sync server config

Return type

dict[str, Any] | list[dict[str, Any]]

delete_sharing_group(*sharing_group*)

Delete a sharing group: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/deleteSharingGroup>

Parameters

sharing_group (*MISPSharingGroup* | int | str | UUID) – sharing group to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_sighting(*sighting*)

Delete a sighting from a MISP instance: <https://www.misp-project.org/openapi/#tag/Sightings/operation/deleteSighting>

Parameters

sighting (*MISPSighting* | int | str | UUID) – sighting to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_tag(*tag*)

Delete a tag from a MISP instance: <https://www.misp-project.org/openapi/#tag/Tags/operation/deleteTag>

Parameters

tag (*MISPTag* | int | str | UUID) – tag to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_user(*user*)

Delete a user by id: <https://www.misp-project.org/openapi/#tag/Users/operation/deleteUser>

Parameters

user (*MISPUser* | int | str | UUID) – user to delete

Return type

dict[str, Any] | list[dict[str, Any]]

delete_user_setting(*user_setting*, *user=None*)

Delete a user setting: <https://www.misp-project.org/openapi/#tag/UserSettings/operation/deleteUserSettingById>

Parameters

- **user_setting** (str) – name of user setting
- **user** (*MISPUser* | int | str | UUID | None) – user

Return type

dict[str, Any] | list[dict[str, Any]]

property describe_types_local: dict[str, Any] | list[dict[str, Any]]

Returns the content of describe types from the package

property describe_types_remote: dict[str, Any] | list[dict[str, Any]]

Returns the content of describe types from the remote instance

direct_call(url, data=None, params={}, kw_params={})

Very lightweight call that posts a data blob (python dictionary or json string) on the URL

Parameters

- **url** (str) – URL to post to
- **data** (dict[str, Any] | None) – data to post
- **params** (Mapping[str, Any]) – dict with parameters for request
- **kw_params** (Mapping[str, Any]) – dict with keyword parameters for request

Return type

Any

disable_decaying_model(decaying_model)

Disable a decaying Model

Return type

dict[str, Any] | list[dict[str, Any]]

disable_feed(feed, pythonify=False)

Disable a feed: <https://www.misp-project.org/openapi/#tag/Feeds/operation/disableFeed>

Parameters

- **feed** (*MISPFeed* | int | str | UUID) – feed to disable
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPFeed*

disable_feed_cache(feed, pythonify=False)

Disable the caching of a feed

Parameters

- **feed** (*MISPFeed* | int | str | UUID) – feed to disable caching
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPFeed*

disable_noticelist(noticelist)

Disable a noticelist by id

Parameters

noticelist (*MISPNoticelist* | int | str | UUID) – Noticelist to disable

Return type

dict[str, Any] | list[dict[str, Any]]

disable_tag(tag, pythonify=False)

Disable a tag

Parameters

- **tag** (*MISPTag*) – tag to disable
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPTag***disable_taxonomy**(taxonomy)Disable a taxonomy: <https://www.misp-project.org/openapi/#tag/Taxonomies/operation/disableTaxonomy>**Parameters****taxonomy** (*MISPTaxonomy* | int | str | UUID) – taxonomy to disable**Return type**

dict[str, Any] | list[dict[str, Any]]

disable_taxonomy_tags(taxonomy)

Disable all the tags of a taxonomy

Parameters**taxonomy** (*MISPTaxonomy* | int | str | UUID) – taxonomy with tags to disable**Return type**

dict[str, Any] | list[dict[str, Any]]

disable_warninglist(warninglist)

Disable a warninglist

Parameters**warninglist** (*MISPWarninglist* | int | str | UUID) – warninglist to disable**Return type**

dict[str, Any] | list[dict[str, Any]]

discard_attribute_proposal(proposal)

Discard a proposal. You cannot modify an existing proposal, only accept/discard

Parameters**proposal** (*MISPShadowAttribute* | int | str | UUID) – attribute proposal to discard**Return type**

dict[str, Any] | list[dict[str, Any]]

discard_event_delegation(delegation, pythonify=False)

Discard the delegation of an event

Parameters

- **delegation** (*MISPEventDelegation* | int | str) – event delegation to discard
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | list[dict[str, Any]]

discard_user_registration(*registration*)

Discard a user registration

Parameters

registration (*MISPInbox* | int | str | UUID) – the registration to discard

Return type

dict[str, Any] | list[dict[str, Any]]

enable_decaying_model(*decaying_model*)

Enable a decaying Model

Return type

dict[str, Any] | list[dict[str, Any]]

enable_feed(*feed*, *pythonify=False*)

Enable a feed; fetching it will create event(s): <https://www.misp-project.org/openapi/#tag/Feeds/operation/enableFeed>

Parameters

- **feed** (*MISPFeed* | int | str | UUID) – feed to enable
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPFeed*

enable_feed_cache(*feed*, *pythonify=False*)

Enable the caching of a feed

Parameters

- **feed** (*MISPFeed* | int | str | UUID) – feed to enable caching
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPFeed*

enable_noticelist(*noticelist*)

Enable a noticelist by id: <https://www.misp-project.org/openapi/#tag/Noticelists/operation/toggleEnableNoticelist>

Parameters

noticelist (*MISPNoticelist* | int | str | UUID) – Noticelist to enable

Return type

dict[str, Any] | list[dict[str, Any]]

enable_tag(*tag*, *pythonify=False*)

Enable a tag

Parameters

- **tag** (*MISPTag*) – tag to enable
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPTag*

enable_taxonomy(*taxonomy*)

Enable a taxonomy: <https://www.misp-project.org/openapi/#tag/Taxonomies/operation/enableTaxonomy>

Parameters

taxonomy (*MISPTaxonomy* | int | str | UUID) – taxonomy to enable

Return type

dict[str, Any] | list[dict[str, Any]]

enable_taxonomy_tags(*taxonomy*)

Enable all the tags of a taxonomy. NOTE: this is automatically done when you call enable_taxonomy

Parameters

taxonomy (*MISPTaxonomy* | int | str | UUID) – taxonomy with tags to enable

Return type

dict[str, Any] | list[dict[str, Any]]

enable_warninglist(*warninglist*)

Enable a warninglist

Parameters

warninglist (*MISPWarninglist* | int | str | UUID) – warninglist to enable

Return type

dict[str, Any] | list[dict[str, Any]]

enrich_attribute(*attribute*, *enrich_with*)

Enrich an attribute with data from one or more module.

Parameters

- **attribute** (*MISPAttribute* | int | str | UUID) – attribute to enrich
- **enrich_with** (str | list[str]) – module name or list of module names to use for enrichment

Return type

dict[str, Any]

enrich_event(*event*, *enrich_with*)

Enrich an event with data from one or more module.

Parameters

- **event** (*MISPEvent* | int | str | UUID) – event to enrich
- **enrich_with** (str | list[str]) – module name or list of module names to use for enrichment

Return type

dict[str, Any]

event_blocklists(*pythonify=False*)

Get all the blocklisted events

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[*MISPEventBlocklist*] | list[dict[str, Any]]

event_delegations(*pythonify=False*)

Get all the event delegations

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPEventDelegation] | list[dict[str, Any]]

event_exists(*event*)

Fast check if event exists.

Parameters

event (MISPEvent | int | str | UUID) – Event to check

Return type

bool

events(*pythonify=False*)

Get all the events from the MISP instance: <https://www.misp-project.org/openapi/#tag/Events/operation/getEvents>

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPEvent] | list[dict[str, Any]]

feeds(*pythonify=False*)

Get the list of existing feeds: <https://www.misp-project.org/openapi/#tag/Feeds/operation/getFeeds>

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPFeed] | list[dict[str, Any]]

fetch_feed(*feed*)

Fetch one single feed by id: <https://www.misp-project.org/openapi/#tag/Feeds/operation/fetchFromFeed>

Parameters

feed (MISPFeed | int | str | UUID) – feed to fetch

Return type

dict[str, Any] | list[dict[str, Any]]

fork_galaxy_cluster(*galaxy, galaxy_cluster, pythonify=False*)

Forks an existing galaxy cluster, creating a new one with matching attributes

Parameters

- **galaxy** (MISPGalaxy | int | str | UUID) – The galaxy (or galaxy ID) where the cluster you want to fork resides
- **galaxy_cluster** (MISPGalaxyCluster) – The galaxy cluster you wish to fork
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | MISPGalaxyCluster

freetext(*event*, *string*, *adhereToWarninglists=False*, *distribution=None*, *returnMetaAttributes=False*, *pythonify=False*, ***kwargs*)

Pass a text to the freetext importer

Parameters

- **event** (*MISPEvent* | int | str | UUID) – event
- **string** (str) – query
- **adhereToWarninglists** (bool | str) – flag
- **distribution** (int | None) – distribution == -1 means recipient decides
- **returnMetaAttributes** (bool) – flag
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output
- **kwargs** – kwargs passed to prepare_request

Return type

dict[str, Any] | list[*MISPAttribute*] | list[dict[str, Any]]

galaxies(*withCluster=False*, *pythonify=False*)

Get all the galaxies: <https://www.misp-project.org/openapi/#tag/Galaxies/operation/getGalaxies>

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[*MISPGalaxy*] | list[dict[str, Any]]

get_all_functions(*not_implemented=False*)

Get all methods available via the API, including ones that are not implemented.

Return type

list[str]

get_analyst_data(*analyst_data*, *pythonify=False*)

Get an analyst data from a MISP instance

Parameters

- **analyst_data** (*MISPEvent* | int | str | UUID) – analyst data to get
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | *MISPNote* | *MISPOpinion* | *MISPRelationship*

get_attribute(*attribute*, *pythonify=False*)

Get an attribute from a MISP instance: <https://www.misp-project.org/openapi/#tag/Attributes/operation/getAttributeById>

Parameters

- **attribute** (*MISPAttribute* | int | str | UUID) – attribute to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPAttribute*

get_attribute_proposal(*proposal*, *pythonify=False*)

Get an attribute proposal

Parameters

- **proposal** (*MISPShadowAttribute* | int | str | UUID) – proposal to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPShadowAttribute*

get_community(*community*, *pythonify=False*)

Get a community by id from a MISP instance

Parameters

- **community** (*MISPCommunity* | int | str | UUID) – community to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPCommunity*

get_correlation_exclusion(*correlation_exclusion*, *pythonify=False*)

Get a correlation exclusion by ID

Parameters

- **correlation_exclusion** (*MISPCorrelationExclusion* | int | str | UUID) – Correlation exclusion to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPCorrelationExclusion*

get_event(*event*, *deleted=False*, *extended=False*, *pythonify=False*)

Get an event from a MISP instance. Includes collections like Attribute, EventReport, Feed, Galaxy, Object, Tag, etc. so the response size may be large : <https://www.misp-project.org/openapi/#tag/Events/operation/getEventById>

Parameters

- **event** (*MISPEvent* | int | str | UUID) – event to get
- **deleted** (bool | int | list[int]) – whether to include soft-deleted attributes
- **extended** (bool | int) – whether to get extended events
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | *MISPEvent*

get_event_report(*event_report*, *pythonify=False*)

Get an event report from a MISP instance

Parameters

- **event_report** (*MISPEventReport* | int | str | UUID) – event report to get
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return typedict[str, Any] | *MISPEventReport***get_event_reports**(*event_id*, *pythonify=False*)

Get event report from a MISP instance that are attached to an event ID

Parameters

- **event_id** (int | str) – event id to get the event reports for
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output.

Return typedict[str, Any] | list[*MISPEventReport*] | list[dict[str, Any]]**get_feed**(*feed*, *pythonify=False*)Get a feed by id: <https://www.misp-project.org/openapi/#tag/Feeds/operation/getFeedById>**Parameters**

- **feed** (*MISPFeed* | int | str | UUID) – feed to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPFeed***get_galaxy**(*galaxy*, *withCluster=False*, *pythonify=False*)Get a galaxy by id: <https://www.misp-project.org/openapi/#tag/Galaxies/operation/getGalaxyById>**Parameters**

- **galaxy** (*MISPGalaxy* | int | str | UUID) – galaxy to get
- **withCluster** (bool) – Include the clusters associated with the galaxy
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPGalaxy***get_galaxy_cluster**(*galaxy_cluster*, *pythonify=False*)

Gets a specific galaxy cluster

Parameters

- **galaxy_cluster** (*MISPGalaxyCluster* | int | str | UUID) – The MISPGalaxyCluster you want to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPGalaxyCluster***get_new_authkey**(*user='me'*)Get a new authorization key for a specific user, defaults to user doing the call: <https://www.misp-project.org/openapi/#tag/AuthKeys/operation/addAuthKey>**Parameters****user** (*MISPUser* | int | str | UUID) – The owner of the key**Return type**

str

get_note(*note*, *pythonify=False*)

Get a note from a MISP instance

Parameters

- **note** (*MISPNote*) – note to get
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output.
Warning: it might use a lot of RAM

Return type

dict[str, Any] | *MISPNote*

get_noticelist(*noticelist*, *pythonify=False*)

Get a noticelist by id: <https://www.misp-project.org/openapi/#tag/Noticelists/operation/getNoticelistById>

Parameters

- **noticelist** – Noticelist to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPNoticelist*

get_object(*misp_object*, *pythonify=False*)

Get an object from the remote MISP instance: <https://www.misp-project.org/openapi/#tag/Objects/operation/getObjectById>

Parameters

- **misp_object** (*MISPObject* | int | str | UUID) – object to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPObject*

get_object_template(*object_template*, *pythonify=False*)

Gets the full object template

Parameters

- **object_template** (*MISPObjectTemplate* | int | str | UUID) – template or ID to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPObjectTemplate*

get_opinion(*opinion*, *pythonify=False*)

Get an opinion from a MISP instance

Parameters

- **opinion** (*MISPOpinion*) – opinion to get
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output.
Warning: it might use a lot of RAM

Return type

dict[str, Any] | *MISPOpinion*

get_organisation(*organisation*, *pythonify=False*)

Get an organisation by id: <https://www.misp-project.org/openapi/#tag/Organisations/operation/getOrganisationById>

Parameters

- **organisation** (*MISPOrganisation* | int | str | UUID) – organization to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPOrganisation*

get_raw_object_template(*uuid_or_name*)

Get a row template. It needs to be present on disk on the MISP instance you're connected to. The response of this method can be passed to MISPObject(<name>, misp_objects_template_custom=<response>)

Return type

dict[str, Any] | list[dict[str, Any]]

get_relationship(*relationship*, *pythonify=False*)

Get a relationship from a MISP instance

Parameters

- **relationship** (*MISPRelationship*) – relationship to get
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | *MISPRelationship*

get_server_setting(*setting*)

Get a setting from the MISP instance

Parameters

setting (str) – server setting name

Return type

dict[str, Any] | list[dict[str, Any]]

get_sharing_group(*sharing_group*, *pythonify=False*)

Get a sharing group: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/getSharingGroupById>

Parameters

- **sharing_group** (*MISPSharingGroup* | int | str | UUID) – sharing group to find
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPSharingGroup*

get_sync_config(*pythonify=False*)

Get the sync server config. WARNING: This method only works if the user calling it is a sync user

Parameters

pythonify (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPServer*

get_tag(tag, pythonify=False)

Get a tag by id: <https://www.misp-project.org/openapi/#tag/Tags/operation/getTagById>

Parameters

- **tag** (*MISPTag* | int | str | UUID) – tag to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPTag*

get_taxonomy(taxonomy, pythonify=False)

Get a taxonomy by id or namespace from a MISP instance: <https://www.misp-project.org/openapi/#tag/Taxonomies/operation/getTaxonomyById>

Parameters

- **taxonomy** (*MISPTaxonomy* | int | str | UUID) – taxonomy to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPTaxonomy*

get_user(user='me', pythonify=False, expanded=False)

Get a user by id: <https://www.misp-project.org/openapi/#tag/Users/operation/getUsers>

Parameters

- **user** (*MISPUser* | int | str | UUID) – user to get; *me* means the owner of the API key doing the query
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output
- **expanded** (bool) – Also returns a MISPRole and a MISPUserSetting. Only taken in account if pythonify is True.

Return type

dict[str, Any] | *MISPUser* | tuple[*MISPUser*, *MISPRole*, list[*MISPUserSetting*]]

get_user_setting(user_setting, user=None, pythonify=False)

Get a user setting: <https://www.misp-project.org/openapi/#tag/UserSettings/operation/getUserSettingById>

Parameters

- **user_setting** (str) – name of user setting
- **user** (*MISPUser* | int | str | UUID | None) – user
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPUserSetting*

get_warninglist(warninglist, pythonify=False)

Get a warninglist by id: <https://www.misp-project.org/openapi/#tag/Warninglists/operation/getWarninglistById>

Parameters

- **warninglist** (*MISPWarninglist* | int | str | UUID) – warninglist to get
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPWarninglist***get_workers()**

Get all the workers

Return type

dict[str, Any] | list[dict[str, Any]]

import_server(server, pythonify=False)

Import a sync server config received from get_sync_config

Parameters

- **server** (*MISPServer*) – sync server config
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPServer***kill_all_workers()**

Kill all the workers

Return type

dict[str, Any] | list[dict[str, Any]]

load_default_feeds()

Load all the default feeds.

Return type

dict[str, Any] | list[dict[str, Any]]

property misp_instance_version: dict[str, Any] | list[dict[str, Any]]

Returns the version of the instance.

property misp_instance_version_master: dict[str, Any] | list[dict[str, Any]]

Get the most recent version from github

noticelists(pythonify=False)Get all the noticelists: <https://www.misp-project.org/openapi/#tag/Noticelists/operation/getNoticelists>**Parameters**

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return typedict[str, Any] | list[*MISPNoticelist*] | list[dict[str, Any]]**object_exists(misp_object)**

Fast check if object exists.

Parameters

misp_object (*MISPObject* | int | str | UUID) – Attribute to check

Return type

bool

object_templates(pythonify=False)

Get all the object templates

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPObjectTemplate] | list[dict[str, Any]]

organisation_blocklists(*pythonify=False*)

Get all the blocklisted organisations

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPOrganisationBlocklist] | list[dict[str, Any]]

organisation_exists(*organisation*)

Fast check if organisation exists.

Parameters

organisation (MISPOrganisation | int | str | UUID) – Organisation to check

Return type

bool

organisations(*scope='local', search=None, pythonify=False*)

Get all the organisations: <https://www.misp-project.org/openapi/#tag/Organisations/operation/getOrganisations>

Parameters

- **scope** (str) – scope of organizations to get
- **search** (str | None) – The search to make against the list of organisations
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPOrganisation] | list[dict[str, Any]]

publish(*event, alert=False*)

Publish the event with one single HTTP POST: <https://www.misp-project.org/openapi/#tag/Events/operation/publishEvent>

Parameters

- **event** (MISPEvent | int | str | UUID) – event to publish
- **alert** (bool) – whether to send an email. The default is to not send a mail as it is assumed this method is called on update.

Return type

dict[str, Any] | list[dict[str, Any]]

publish_galaxy_cluster(*galaxy_cluster*)

Publishes a galaxy cluster: <https://www.misp-project.org/openapi/#tag/Galaxy-Clusters/operation/publishGalaxyCluster>

Parameters

galaxy_cluster (MISPGalaxyCluster | int | str | UUID) – The galaxy cluster you wish to publish

Return type

dict[str, Any] | list[dict[str, Any]]

push_event_to_ZMQ(*event*)

Force push an event by id on ZMQ

Parameters**event** (*MISPEvent* | int | str | UUID) – the event to push**Return type**

dict[str, Any] | list[dict[str, Any]]

property pymisp_version_main: dict[str, Any] | list[dict[str, Any]]

Get the most recent version of PyMISP from github

property pymisp_version_master: dict[str, Any] | list[dict[str, Any]]

PyMISP version as defined in the main repository

property recommended_pymisp_version: dict[str, Any] | list[dict[str, Any]]

Returns the recommended API version from the server

remote_acl(*debug_type='findMissingFunctionNames'*)

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

Parameters**debug_type** (str) – printAllFunctionNames, findMissingFunctionNames, or printRoleAccess**Return type**

dict[str, Any] | list[dict[str, Any]]

remove_org_from_sharing_group(*sharing_group, organisation*)Remove an organisation from a sharing group: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/removeOrganisationFromSharingGroup>**Parameters**

- **sharing_group** (*MISPSharingGroup* | int | str | UUID) – Sharing group’s local instance ID, or Sharing group’s global UUID
- **organisation** (*MISPOrganisation* | int | str | UUID) – Organisation’s local instance ID, or Organisation’s global UUID, or Organisation’s name as known to the current instance

Return type

dict[str, Any] | list[dict[str, Any]]

remove_server_from_sharing_group(*sharing_group, server*)Remove a server from a sharing group: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/removeServerFromSharingGroup>**Parameters**

- **sharing_group** (*MISPSharingGroup* | int | str | UUID) – Sharing group’s local instance ID, or Sharing group’s global UUID
- **server** (*MISPServer* | int | str | UUID) – Server’s local instance ID, or URL of the Server, or Server’s name as known to the current instance

Return type

dict[str, Any] | list[dict[str, Any]]

request_community_access(*community, requestor_email_address=None, requestor_gpg_key=None, requestor_organisation_name=None, requestor_organisation_uuid=None, requestor_organisation_description=None, message=None, sync=False, anonymise_requestor_server=False, mock=False*)

Request the access to a community

Parameters

- **community** (MISPCommunity | int | str | UUID) – community to request access
- **requestor_email_address** (str | None) – requestor email
- **requestor_gpg_key** (str | None) – requestor key
- **requestor_organisation_name** (str | None) – requestor org name
- **requestor_organisation_uuid** (str | None) – requestor org ID
- **requestor_organisation_description** (str | None) – requestor org desc
- **message** (str | None) – requestor message
- **sync** (bool) – synchronize flag
- **anonymise_requestor_server** (bool) – anonymise flag
- **mock** (bool) – mock flag

Return type

dict[str, Any] | list[dict[str, Any]]

restart_dead_workers()

Restart the dead workers

Return type

dict[str, Any] | list[dict[str, Any]]

restart_workers()

Restart all the workers

Return type

dict[str, Any] | list[dict[str, Any]]

restore_attribute(*attribute, pythonify=False*)

Restore a soft deleted attribute from a MISP instance: <https://www.misp-project.org/openapi/#tag/Attributes/operation/restoreAttribute>

Parameters

attribute (MISPAttribute | int | str | UUID) – attribute to restore

Return type

dict[str, Any] | MISPAttribute

roles(*pythonify=False*)

Get the existing roles

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPRole] | list[dict[str, Any]]

```
search(controller='events', return_format='json', limit=None, page=None, value=None,
        type_attribute=None, category=None, org=None, tags=None, event_tags=None, quick_filter=None,
        quickFilter=None, date_from=None, date_to=None, eventid=None, with_attachments=None,
        withAttachments=None, metadata=None, uuid=None, publish_timestamp=None, last=None,
        timestamp=None, published=None, enforce_warninglist=None, enforceWarninglist=None,
        to_ids=None, deleted=None, include_event_uuid=None, includeEventUuid=None,
        include_event_tags=None, includeEventTags=None, event_timestamp=None,
        sg_reference_only=None, eventinfo=None, searchall=None, requested_attributes=None,
        include_context=None, includeContext=None, headerless=None, include_sightings=None,
        includeSightings=None, include_correlations=None, includeCorrelations=None,
        include_decay_score=None, includeDecayScore=None, object_name=None,
        exclude_decayed=None, sharinggroup=None, pythonify=False, **kwargs)
```

Search in the MISP instance

Parameters

- **controller** (str) – Controller to search on, it can be *events*, *objects*, *attributes*. The response will either be a list of events, objects, or attributes. Reference documentation for each controller:
 - events: <https://www.misp-project.org/openapi/#tag/Events/operation/restSearchEvents>
 - attributes: <https://www.misp-project.org/openapi/#tag/Attributes/operation/restSearchAttributes>
 - objects: N/A
- **return_format** (str) – Set the return format of the search (Currently supported: json, xml, openioc, suricata, snort - more formats are being moved to restSearch with the goal being that all searches happen through this API). Can be passed as the first parameter after restSearch or via the JSON payload.
- **limit** (int | None) – Limit the number of results returned, depending on the scope (for example 10 attributes or 10 full events).
- **page** (int | None) – If a limit is set, sets the page to be returned. page 3, limit 100 will return records 201->300).
- **value** (TypeVar(SearchParameterTypes, str, List[str | int], Dict[str, str | int]) | None) – Search for the given value in the attributes' value field.
- **type_attribute** (TypeVar(SearchParameterTypes, str, List[str | int], Dict[str, str | int]) | None) – The attribute type, any valid MISP attribute type is accepted.
- **category** (TypeVar(SearchParameterTypes, str, List[str | int], Dict[str, str | int]) | None) – The attribute category, any valid MISP attribute category is accepted.
- **org** (TypeVar(SearchParameterTypes, str, List[str | int], Dict[str, str | int]) | None) – Search by the creator organisation by supplying the organisation identifier.
- **tags** (TypeVar(SearchParameterTypes, str, List[str | int], Dict[str, str | int]) | None) – Tags to search or to exclude. You can pass a list, or the output of *build_complex_query*
- **event_tags** (TypeVar(SearchParameterTypes, str, List[str | int], Dict[str, str | int]) | None) – Tags to search or to exclude at the event level. You can pass a list, or the output of *build_complex_query*
- **quick_filter** (str | None) – The string passed to this field will ignore all of the other arguments. MISP will return an xml / json (depending on the header sent) of all events that

have a sub-string match on value in the event info, event orgc, or any of the attribute value1 / value2 fields, or in the attribute comment.

- **date_from** (datetime | date | int | str | float | None) – Events with the date set to a date after the one specified. This filter will use the date of the event.
- **date_to** (datetime | date | int | str | float | None) – Events with the date set to a date before the one specified. This filter will use the date of the event.
- **eventid** (TypeVar(SearchType, str, int) | None) – The events that should be included / excluded from the search
- **with_attachments** (bool | None) – If set, encodes the attachments / zipped malware samples as base64 in the data field within each attribute
- **metadata** (bool | None) – Only the metadata (event, tags, relations) is returned, attributes and proposals are omitted.
- **uuid** (str | None) – Restrict the results by uuid.
- **publish_timestamp** (None | datetime | date | int | str | float | tuple[datetime | date | int | str | float | None, datetime | date | int | str | float | None]) – Restrict the results by the last publish timestamp (newer than).
- **timestamp** (None | datetime | date | int | str | float | tuple[datetime | date | int | str | float | None, datetime | date | int | str | float | None]) – Restrict the results by the timestamp (last edit). Any event with a timestamp newer than the given timestamp will be returned. In case you are dealing with /attributes as scope, the attribute’s timestamp will be used for the lookup. The input can be a timestamp or a short-hand time description (7d or 24h for example). You can also pass a list with two values to set a time range (for example [“14d”, “7d”]).
- **published** (bool | None) – Set whether published or unpublished events should be returned. Do not set the parameter if you want both.
- **enforce_warninglist** (bool | None) – Remove any attributes from the result that would cause a hit on a warninglist entry.
- **to_ids** (TypeVar(ToIDSType, str, int, bool) | list[TypeVar(ToIDSType, str, int, bool)]) | None) – By default all attributes are returned that match the other filter parameters, regardless of their to_ids setting. To restrict the returned data set to to_ids only attributes set this parameter to 1. 0 for the ones with to_ids set to False.
- **deleted** (str | None) – If this parameter is set to 1, it will only return soft-deleted attributes. [“0”, “1”] will return the active ones as well as the soft-deleted ones.
- **include_event_uuid** (bool | None) – Instead of just including the event ID, also include the event UUID in each of the attributes.
- **include_event_tags** (bool | None) – Include the event level tags in each of the attributes.
- **event_timestamp** (datetime | date | int | str | float | None) – Only return attributes from events that have received a modification after the given timestamp.
- **sg_reference_only** (bool | None) – If this flag is set, sharing group objects will not be included, instead only the sharing group ID is set.
- **eventinfo** (str | None) – Filter on the event’s info field.
- **searchall** (bool | None) – Search for a full or a substring (delimited by % for substrings) in the event info, event tags, attribute tags, attribute values or attribute comment fields.

- **requested_attributes** (str | None) – [CSV only] Select the fields that you wish to include in the CSV export. By setting event level fields additionally, includeContext is not required to get event metadata.
- **include_context** (bool | None) – [Attribute only] Include the event data with each attribute. [CSV output] Add event level metadata in every line of the CSV.
- **headerless** (bool | None) – [CSV Only] The CSV created when this setting is set to true will not contain the header row.
- **include_sightings** (bool | None) – [JSON Only - Attribute] Include the sightings of the matching attributes.
- **include_decay_score** (bool | None) – Include the decay score at attribute level.
- **include_correlations** (bool | None) – [JSON Only - attribute] Include the correlations of the matching attributes.
- **object_name** (str | None) – [objects controller only] Search for objects with that name
- **exclude_decayed** (bool | None) – [attributes controller only] Exclude the decayed attributes from the response
- **sharinggroup** (int | list[int] | None) – Filter by sharing group ID(s)
- **pythonify** (bool | None) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | str | list[MISPEvent | MISPAtribute | MISPObject] | list[dict[str, Any]]

Deprecated:

Parameters

- **quickFilter** (str | None) – synonym for quick_filter
- **withAttachments** (bool | None) – synonym for with_attachments
- **last** (None | datetime | date | int | str | float | tuple[datetime | date | int | str | float | None, datetime | date | int | str | float | None]) – synonym for publish_timestamp
- **enforceWarninglist** (bool | None) – synonym for enforce_warninglist
- **includeEventUuid** (bool | None) – synonym for include_event_uuid
- **includeEventTags** (bool | None) – synonym for include_event_tags
- **includeContext** (bool | None) – synonym for include_context

search_feeds(value=None, pythonify=False)

Search in the feeds cached on the servers

Return type

dict[str, Any] | list[MISPFeed] | list[dict[str, Any]]

search_galaxy(value, withCluster=False, pythonify=False)

Text search to find a matching galaxy name, namespace, description, or uuid.

Return type

dict[str, Any] | list[MISPGalaxy] | list[dict[str, Any]]

search_galaxy_clusters(*galaxy, context='all', searchall=None, pythonify=False*)

Searches the galaxy clusters within a specific galaxy: <https://www.misp-project.org/openapi/#tag/Galaxy-Clusters/operation/getGalaxyClusters> and <https://www.misp-project.org/openapi/#tag/Galaxy-Clusters/operation/getGalaxyClusterById>

Parameters

- **galaxy** (*MISPGalaxy* | int | str | UUID) – The MISPGalaxy you wish to search in
- **context** (str) – The context of how you want to search within the **galaxy_**
- **searchall** (str | None) – The search you want to make against the galaxy and context
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | list[MISPGalaxyCluster] | list[dict[str, Any]]

search_index(*all=None, attribute=None, email=None, published=None, hasproposal=None, eventid=None, tags=None, date_from=None, date_to=None, eventinfo=None, threatlevel=None, distribution=None, analysis=None, org=None, timestamp=None, publish_timestamp=None, sharinggroup=None, minimal=None, sort=None, desc=None, limit=None, page=None, pythonify=None*)

Search event metadata shown on the event index page. Using ! in front of a value means NOT, except for parameters `date_from`, `date_to` and `timestamp` which cannot be negated. Criteria are AND-ed together; values in lists are OR-ed together. Return matching events with metadata but no attributes or objects; also see `minimal` parameter.

Parameters

- **all** (str | None) – Search for a full or a substring (delimited by % for substrings) in the event info, event tags, attribute tags, attribute values or attribute comment fields.
- **attribute** (str | None) – Filter on attribute's value.
- **email** (str | None) – Filter on user's email.
- **published** (bool | None) – Set whether published or unpublished events should be returned. Do not set the parameter if you want both.
- **hasproposal** (bool | None) – Filter for events containing proposal(s).
- **eventid** (TypeVar(SearchType, str, int) | None) – The events that should be included / excluded from the search
- **tags** (TypeVar(SearchParameterTypes, str, List[str | int], Dict[str, str | int]) | None) – Tags to search or to exclude. You can pass a list, or the output of `build_complex_query`
- **date_from** (datetime | date | int | str | float | None) – Events with the date set to a date after the one specified. This filter will use the date of the event.
- **date_to** (datetime | date | int | str | float | None) – Events with the date set to a date before the one specified. This filter will use the date of the event.
- **eventinfo** (str | None) – Filter on the event's info field.
- **threatlevel** (list[TypeVar(SearchType, str, int)] | None) – Threat level(s) (1,2,3,4) | list
- **distribution** (list[TypeVar(SearchType, str, int)] | None) – Distribution level(s) (0,1,2,3) | list

- **analysis** (`list[TypeVar(SearchType, str, int)] | None`) – Analysis level(s) (0,1,2) | list
- **org** (`TypeVar(SearchParameterTypes, str, List[str | int], Dict[str, str | int]) | None`) – Search by the creator organisation by supplying the organisation identifier.
- **timestamp** (`None | datetime | date | int | str | float | tuple[datetime | date | int | str | float | None, datetime | date | int | str | float | None]`) – Restrict the results by the timestamp (last edit). Any event with a timestamp newer than the given timestamp will be returned. In case you are dealing with /attributes as scope, the attribute’s timestamp will be used for the lookup.
- **publish_timestamp** (`None | datetime | date | int | str | float | tuple[datetime | date | int | str | float | None, datetime | date | int | str | float | None]`) – Filter on event’s publish timestamp.
- **sharinggroup** (`list[TypeVar(SearchType, str, int)] | None`) – Restrict by a sharing group | list
- **minimal** (`bool | None`) – Return only event ID, UUID, timestamp, sighting_timestamp and published.
- **sort** (`str | None`) – The field to sort the events by, such as ‘id’, ‘date’, ‘attribute_count’.
- **desc** (`bool | None`) – Whether to sort events ascending (default) or descending.
- **limit** (`int | None`) – Limit the number of events returned
- **page** (`int | None`) – If a limit is set, sets the page to be returned. page 3, limit 100 will return records 201->300).
- **pythonify** (`bool | None`) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

`dict[str, Any] | list[MISPEvent] | list[dict[str, Any]]`

search_logs(*limit=None, page=None, log_id=None, title=None, created=None, model=None, action=None, user_id=None, change=None, email=None, org=None, description=None, ip=None, pythonify=False*)

Search in logs

Note: to run substring queries simply append/prepend/encapsulate the search term with %

Parameters

- **limit** (`int | None`) – Limit the number of results returned, depending on the scope (for example 10 attributes or 10 full events).
- **page** (`int | None`) – If a limit is set, sets the page to be returned. page 3, limit 100 will return records 201->300).
- **log_id** (`int | None`) – Log ID
- **title** (`str | None`) – Log Title
- **created** (`datetime | date | int | str | float | None`) – Creation timestamp
- **model** (`str | None`) – Model name that generated the log entry
- **action** (`str | None`) – The thing that was done
- **user_id** (`int | None`) – ID of the user doing the action
- **change** (`str | None`) – Change that occurred

- **email** (str | None) – Email of the user
- **org** (str | None) – Organisation of the User doing the action
- **description** (str | None) – Description of the action
- **ip** (str | None) – Origination IP of the User doing the action
- **pythonify** (bool | None) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPLog] | list[dict[str, Any]]

search_sightings(context=None, context_id=None, type_sighting=None, date_from=None, date_to=None, publish_timestamp=None, last=None, org=None, source=None, include_attribute=None, include_event_meta=None, pythonify=False)

Search sightings

Parameters

- **context** (str | None) – The context of the search. Can be either “attribute”, “event”, or nothing (will then match on events and attributes).
- **context_id** (TypeVar(SearchType, str, int) | None) – Only relevant if context is either “attribute” or “event”. Then it is the relevant ID.
- **type_sighting** (str | None) – Type of sighting
- **date_from** (datetime | date | int | str | float | None) – Events with the date set to a date after the one specified. This filter will use the date of the event.
- **date_to** (datetime | date | int | str | float | None) – Events with the date set to a date before the one specified. This filter will use the date of the event.
- **publish_timestamp** (None | datetime | date | int | str | float | tuple[datetime | date | int | str | float | None, datetime | date | int | str | float | None]) – Restrict the results by the last publish timestamp (newer than).
- **org** (TypeVar(SearchType, str, int) | None) – Search by the creator organisation by supplying the organisation identifier.
- **source** (str | None) – Source of the sighting
- **include_attribute** (bool | None) – Include the attribute.
- **include_event_meta** (bool | None) – Include the meta information of the event.

Return type

dict[str, Any] | list[dict[str, MISPEvent | MISPAtribute | MISPSighting]]

Deprecated:

Parameters

last (None | datetime | date | int | str | float | tuple[datetime | date | int | str | float | None, datetime | date | int | str | float | None]) – synonym for publish_timestamp

Example

```
>>> misp.search_sightings(publish_timestamp='30d') # search sightings for the
↳ last 30 days on the instance
[ ... ]
>>> misp.search_sightings(context='attribute', context_id=6, include_
```

(continues on next page)

(continued from previous page)

```

↳attribute=True) # return list of sighting for attribute 6 along with the_
↳attribute itself
[ ... ]
>>> misp.search_sightings(context='event', context_id=17, include_event_
↳meta=True, org=2) # return list of sighting for event 17 filtered with org id_
↳2

```

search_tags(tagname, strict_tagname=False, pythonify=False)

Search for tags by name: <https://www.misp-project.org/openapi/#tag/Tags/operation/searchTag>

Parameters

- **tag_name** – Name to search, use % for substrings matches.
- **strict_tagname** (bool) – only return tags matching exactly the tag name (so skipping synonyms and cluster's value)

Return type

dict[str, Any] | list[MISPTag] | list[dict[str, Any]]

server_pull(server, event=None)

Initialize a pull from a sync server, optionally limited to one event: <https://www.misp-project.org/openapi/#tag/Servers/operation/pullServer>

Parameters

- **server** (MISPServer | int | str | UUID) – sync server config
- **event** (MISPEvent | int | str | UUID | None) – event

Return type

dict[str, Any] | list[dict[str, Any]]

server_push(server, event=None)

Initialize a push to a sync server, optionally limited to one event: <https://www.misp-project.org/openapi/#tag/Servers/operation/pushServer>

Parameters

- **server** (MISPServer | int | str | UUID) – sync server config
- **event** (MISPEvent | int | str | UUID | None) – event

Return type

dict[str, Any] | list[dict[str, Any]]

server_settings()

Get all the settings from the server

Return type

dict[str, Any] | list[dict[str, Any]]

servers(pythonify=False)

Get the existing servers the MISP instance can synchronise with: <https://www.misp-project.org/openapi/#tag/Servers/operation/getServers>

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPServer] | list[dict[str, Any]]

set_default_role(role)

Set a default role for the new user accounts

Parameters

role (MISPRole | int | str | UUID) – the default role to set

Return type

dict[str, Any] | list[dict[str, Any]]

set_server_setting(setting, value, force=False)

Set a setting on the MISP instance

Parameters

- **setting** (str) – server setting name
- **value** (str | int | bool) – value to set
- **force** (bool) – override value test

Return type

dict[str, Any] | list[dict[str, Any]]

set_user_setting(user_setting, value, user=None, pythonify=False)

Set a user setting: <https://www.misp-project.org/openapi/#tag/UserSettings/operation/setUserSetting>

Parameters

- **user_setting** (str) – name of user setting
- **value** (str | dict[str, Any]) – value to set
- **user** (MISPUser | int | str | UUID | None) – user
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | MISPUserSetting

sharing_group_exists(sharing_group)

Fast check if sharing group exists.

Parameters

sharing_group (MISPSharingGroup | int | str | UUID) – Sharing group to check

Return type

bool

sharing_groups(pythonify=False)

Get the existing sharing groups: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/getSharingGroup>

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPSharingGroup] | list[dict[str, Any]]

sightings(*misp_entity=None, org=None, pythonify=False*)

Get the list of sightings related to a MISPEvent or a MISPAttribute (depending on type of *misp_entity*):
<https://www.misp-project.org/openapi/#tag/Sightings/operation/getSightingsByEventId>

Parameters

- **misp_entity** (*AbstractMISP* | None) – MISP entity
- **org** (*MISPOrganisation* | int | str | UUID | None) – MISP organization
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output.
Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPSighting] | list[dict[str, Any]]

sign_blob(*blob*)

Sign a blob

Parameters

blob (str) – blob to sign

Return type

str

start_worker(*worker_type*)

Start a worker :type worker_type: str :param worker_type: The type of worker, must be one of the following: “default” “email” “scheduler” “cache” “prio” “update”

Return type

dict[str, Any] | list[dict[str, Any]]

stop_worker_by_pid(*worker_pid*)

Stop a worker by its PID (get the pid with `get_workers`) :type worker_pid: str | int :param worker_pid: The pid of the worker to stop

Return type

dict[str, Any] | list[dict[str, Any]]

tag(*misp_entity, tag, local=False, relationship_type=None*)

Tag an event or an attribute.

Parameters

- **misp_entity** (*AbstractMISP* | str | dict[str, Any]) – a MISPEvent, a MISP Attribute, or a UUID
- **tag** (*MISPTag* | str | dict[str, Any]) – tag to add
- **local** (bool) – whether to tag locally
- **relationship_type** (str | None) – Type of relationship between the tag and the attribute or event

Return type

dict[str, Any] | list[dict[str, Any]]

tags(*pythonify=False, **kw_params*)

Get the list of existing tags: <https://www.misp-project.org/openapi/#tag/Tags/operation/getTags>

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPTag]

tags_statistics(*percentage=False, name_sort=False*)

Get tag statistics from the MISP instance

Parameters

- **percentage** (bool) – get percentages
- **name_sort** (bool) – sort by name

Return type

dict[str, Any] | list[dict[str, Any]]

taxonomies(*pythonify=False*)

Get all the taxonomies: <https://www.misp-project.org/openapi/#tag/Taxonomies/operation/getTaxonomies>

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPTaxonomy] | list[dict[str, Any]]

test_server(*server*)

Test if a sync link is working as expected

Parameters

server (MISPServer | int | str | UUID) – sync server config

Return type

dict[str, Any] | list[dict[str, Any]]

toggle_global_pythonify()

Toggle the pythonify variable for the class

Return type

None

toggle_warninglist(*warninglist_id=None, warninglist_name=None, force_enable=None*)

Toggle (enable/disable) the status of a warninglist by id: <https://www.misp-project.org/openapi/#tag/Warninglists/operation/toggleEnableWarninglist>

Parameters

- **warninglist_id** (str | int | list[int] | None) – ID of the WarningList
- **warninglist_name** (str | list[str] | None) – name of the WarningList
- **force_enable** (bool | None) – Force the warning list in the enabled state (does nothing if already enabled) - None means toggle.

Return type

dict[str, Any] | list[dict[str, Any]]

unpublish(*event*)

Unpublish the event with one single HTTP POST: <https://www.misp-project.org/openapi/#tag/Events/operation/unpublishEvent>

Parameters

event (MISPEvent | int | str | UUID) – event to unpublish

Return type

dict[str, Any] | list[dict[str, Any]]

untag(*misp_entity*, *tag*)

Untag an event or an attribute

Parameters

- **misp_entity** (*AbstractMISP* | str | dict[str, Any]) – misp_entity can be a UUID
- **tag** (*MISPTag* | str) – tag to remove

Return type

dict[str, Any] | list[dict[str, Any]]

update_analyst_data(*analyst_data*, *analyst_data_id=None*, *pythonify=False*)

Update an analyst data on a MISP instance

Parameters

- **analyst_data** (*MISPNote* | *MISPOpinion* | *MISPRelationship*) – analyst data to update
- **analyst_data_id** (int | None) – analyst data ID to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPNote* | *MISPOpinion* | *MISPRelationship***update_attribute**(*attribute*, *attribute_id=None*, *pythonify=False*)Update an attribute on a MISP instance: <https://www.misp-project.org/openapi/#tag/Attributes/operation/editAttribute>**Parameters**

- **attribute** (*MISPAttribute*) – attribute to update
- **attribute_id** (int | None) – attribute ID to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPAttribute* | *MISPShadowAttribute***update_attribute_proposal**(*initial_attribute*, *attribute*, *pythonify=False*)

Propose a change for an attribute

Parameters

- **initial_attribute** (*MISPAttribute* | int | str | UUID) – attribute to change
- **attribute** (*MISPAttribute*) – attribute to propose
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return typedict[str, Any] | *MISPShadowAttribute***update_decaying_models**()

Update all the Decaying models

Return type

dict[str, Any] | list[dict[str, Any]]

update_event(*event*, *event_id=None*, *pythonify=False*, *metadata=False*)

Update an event on a MISP instance: <https://www.misp-project.org/openapi/#tag/Events/operation/editEvent>

Parameters

- **event** (*MISPEvent*) – event to update
- **event_id** (int | None) – ID of event to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output
- **metadata** (bool) – Return just event metadata after successful update

Return type

dict[str, Any] | *MISPEvent*

update_event_blocklist(*event_blocklist*, *event_blocklist_id=None*, *pythonify=False*)

Update an event in the blocklist

Parameters

- **event_blocklist** (*MISPEventBlocklist*) – event block list
- **event_blocklist_id** (int | str | UUID | None) – event block list id
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPEventBlocklist*

update_event_report(*event_report*, *event_report_id=None*, *pythonify=False*)

Update an event report on a MISP instance

Parameters

- **event_report** (*MISPEventReport*) – event report to update
- **event_report_id** (int | None) – event report ID to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPEventReport*

update_feed(*feed*, *feed_id=None*, *pythonify=False*)

Update a feed on a MISP instance

Parameters

- **feed** (*MISPFeed*) – feed to update
- **feed_id** (int | None) – feed id
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPFeed*

update_galaxies()

Update all the galaxies: <https://www.misp-project.org/openapi/#tag/Galaxies/operation/updateGalaxies>

Return type

dict[str, Any] | list[dict[str, Any]]

update_galaxy_cluster(*galaxy_cluster*, *pythonify=False*)

Update a custom galaxy cluster: <https://www.misp-project.org/openapi/#tag/Galaxy-Clusters/operation/editGalaxyCluster>

:param *galaxy_cluster*: The MISPGalaxyCluster you wish to update :type *pythonify*: bool :param *pythonify*: Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPGalaxyCluster*

update_galaxy_cluster_relation(*galaxy_cluster_relation*)

Update a galaxy cluster relation

Parameters

galaxy_cluster_relation (*MISPGalaxyClusterRelation*) – The MISPGalaxyClusterRelation to update

Return type

dict[str, Any] | list[dict[str, Any]]

update_misp()

Trigger a server update

Return type

dict[str, Any] | list[dict[str, Any]]

update_note(*note*, *note_id=None*, *pythonify=False*)

Update a note on a MISP instance

Parameters

- **note** (*MISPNote*) – note to update
- **note_id** (int | None) – note ID to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPNote*

update_noticelists()

Update all the noticelists: <https://www.misp-project.org/openapi/#tag/Noticelists/operation/updateNoticelists>

Return type

dict[str, Any] | list[dict[str, Any]]

update_object(*misp_object*, *object_id=None*, *pythonify=False*)

Update an object on a MISP instance

Parameters

- **misp_object** (*MISPObject*) – object to update
- **object_id** (int | None) – ID of object to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPObject*

update_object_templates()

Trigger an update of the object templates

Return type

dict[str, Any] | list[dict[str, Any]]

update_opinion(*opinion*, *opinion_id=None*, *pythonify=False*)

Update an opinion on a MISP instance

Parameters

- **opinion** (*MISPOpinion*) – opinion to update
- **opinion_id** (int | None) – opinion ID to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPOpinion*

update_organisation(*organisation*, *organisation_id=None*, *pythonify=False*)

Update an organisation: <https://www.misp-project.org/openapi/#tag/Organisations/operation/editOrganisation>

Parameters

- **organisation** (*MISPOrganisation*) – organization to update
- **organisation_id** (int | None) – id to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPOrganisation*

update_organisation_blocklist(*organisation_blocklist*, *organisation_blocklist_id=None*, *pythonify=False*)

Update an organisation in the blocklist

Parameters

- **organisation_blocklist** (*MISPOrganisationBlocklist*) – organization block list
- **organisation_blocklist_id** (int | str | UUID | None) – organization block list id
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPOrganisationBlocklist*

update_relationship(*relationship*, *relationship_id=None*, *pythonify=False*)

Update a relationship on a MISP instance

Parameters

- **relationship** (*MISPRelationship*) – relationship to update
- **relationship_id** (int | None) – relationship ID to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPRelationship*

update_role(*role*, *role_id=None*, *pythonify=False*)

Update a role on a MISP instance

Parameters

- **role** (*MISPRole*) – role to update
- **role_id** (int | None) – id to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPRole*

update_server(*server*, *server_id=None*, *pythonify=False*)

Update a server to synchronise with: <https://www.misp-project.org/openapi/#tag/Servers/operation/getServers>

Parameters

- **server** (*MISPServer*) – sync server config
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPServer*

update_sharing_group(*sharing_group*, *sharing_group_id=None*, *pythonify=False*)

Update sharing group parameters: <https://www.misp-project.org/openapi/#tag/Sharing-Groups/operation/editSharingGroup>

Return type

dict[str, Any] | *MISPSharingGroup*

Parameters

sharing_group (*MISPSharingGroup* | dict[str, Any]) – MISP Sharing Group

:param sharing_group_id Sharing group ID :type pythonify: bool :param pythonify: Returns a PyMISP Object instead of the plain json output

update_tag(*tag*, *tag_id=None*, *pythonify=False*)

Edit only the provided parameters of a tag: <https://www.misp-project.org/openapi/#tag/Tags/operation/editTag>

Parameters

- **tag** (*MISPTag*) – tag to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Aram tag_id

tag ID to update

Return type

dict[str, Any] | *MISPTag*

update_taxonomies()

Update all the taxonomies: <https://www.misp-project.org/openapi/#tag/Taxonomies/operation/updateTaxonomies>

Return type

dict[str, Any] | list[dict[str, Any]]

update_user(*user*, *user_id=None*, *pythonify=False*)

Update a user on a MISP instance: <https://www.misp-project.org/openapi/#tag/Users/operation/editUser>

Parameters

- **user** (*MISPUser*) – user to update
- **user_id** (int | None) – id to update
- **pythonify** (bool) – Returns a PyMISP Object instead of the plain json output

Return type

dict[str, Any] | *MISPUser*

update_warninglists()

Update all the warninglists: <https://www.misp-project.org/openapi/#tag/Warninglists/operation/updateWarninglists>

Return type

dict[str, Any] | list[dict[str, Any]]

upload_stix(*path=None*, *data=None*, *version='2'*)

Upload a STIX file to MISP.

Parameters

- **path** (str | Path | BytesIO | StringIO | None) – Path to the STIX on the disk (can be a path-like object, or a pseudofile)
- **data** (str | bytes | None) – stix object
- **version** (str) – Can be 1 or 2

Return type

Response

user_registrations(*pythonify=False*)

Get all the user registrations

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[*MISPInbox*] | list[dict[str, Any]]

user_settings(*pythonify=False*)

Get all the user settings: <https://www.misp-project.org/openapi/#tag/UserSettings/operation/getUserSettings>

Parameters

pythonify (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[*MISPUserSetting*] | list[dict[str, Any]]

users(*search=None*, *organisation=None*, *pythonify=False*)

Get all the users, or a filtered set of users: <https://www.misp-project.org/openapi/#tag/Users/operation/getUsers>

Parameters

- **search** (str | None) – The search to make against the list of users

- **organisation** (int | None) – The ID of an organisation to filter against
- **pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output.
Warning: it might use a lot of RAM

Return type

dict[str, Any] | list[MISPUser] | list[dict[str, Any]]

users_heartbeat()

?

Return type

dict[str, str]

users_statistics(context='data')

Get user statistics from the MISP instance

Parameters**context** (str) – one of 'data', 'orgs', 'users', 'tags', 'attributehistogram', 'sightings', 'galaxy-Matrix'**Return type**

dict[str, Any] | list[dict[str, Any]]

values_in_warninglist(value)

Check if IOC values are in warninglist

Parameters**value** (Iterable[str]) – iterator with values to check**Return type**

dict[str, Any] | list[dict[str, Any]]

property version: dict[str, Any] | list[dict[str, Any]]

Returns the version of PyMISP you're currently using

warninglists(pythonify=False)Get all the warninglists: <https://www.misp-project.org/openapi/#tag/Warninglists/operation/getWarninglists>**Parameters****pythonify** (bool) – Returns a list of PyMISP Objects instead of the plain json output. Warning: it might use a lot of RAM**Return type**

dict[str, Any] | list[MISPWarninglist]

3.2 MISPAbstract

class pymisp.**AbstractMISP**(force_timestamps=False)**property edited:** bool

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

jsonable()

This method is used by the JSON encoder

Return type

dict[str, Any]

set_not_jsonable(*args*)Set `__not_jsonable` to a new list**Return type**

None

to_dict(*json_format=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False, indent=None*)Dump recursively any class of type `MISPAbstract` to a json string**Return type**

str

update_not_jsonable(**args*)Add entries to the `__not_jsonable` list**Return type**

None

3.3 MISPEncode

class pymisp.MISPEncode(**args*, ***kwargs*)**default**(*obj*)Implement this method in a subclass such that it returns a serializable object for `o`, or calls the base implementation (to raise a `TypeError`).For example, to support arbitrary iterators, you could implement `default` like this:

```
def default(self, o):
    try:
        iterable = iter(o)
    except TypeError:
        pass
    else:
        return list(iterable)
```

(continues on next page)

(continued from previous page)

```
# Let the base class default method raise the TypeError
return super().default(o)
```

Return type
dict[str, Any] | str

3.4 MISPEvent

class pymisp.MISPEvent(*describe_types=None, strict_validation=False, force_timestamps=False, **kwargs*)

add_attribute(*type, value, **kwargs*)

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

Return type
MISPAttribute | list[MISPAttribute]

add_attribute_tag(*tag, attribute_identifier*)

Add a tag to an existing attribute. Raise an Exception if the attribute doesn't exist.

Parameters

- **tag** (*MISPTag* | str) – Tag name as a string, MISPTag instance, or dictionary
- **attribute_identifier** (str) – can be an ID, UUID, or the value.

Return type
list[MISPAttribute]

add_cryptographic_key(*parent_type, key_data, type, uuid, fingerprint, timestamp, **kwargs*)

Add a Cryptographic Key. parent_type, key_data, type, uuid, fingerprint, timestamp are required but you can pass all other parameters supported by MISPEventReport

Return type
MISPCryptographicKey

add_event_report(*name, content, **kwargs*)

Add an event report. name and value are required but you can pass all other parameters supported by MISPEventReport

Return type
MISPEventReport

add_galaxy(*galaxy=None, **kwargs*)

Add a galaxy and sub-clusters into an event, either by passing a MISPGalaxy or a dictionary. Supports all other parameters supported by MISPGalaxy

Return type
MISPGalaxy

add_object(*obj=None, **kwargs*)

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

Return type
MISPObject

add_proposal(*shadow_attribute=None, **kwargs*)

Alias for add_shadow_attribute

Return type

MISPShadowAttribute

add_shadow_attribute(*shadow_attribute=None, **kwargs*)

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

Return type

MISPShadowAttribute

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

delete_attribute(*attribute_id*)

Delete an attribute

Parameters

attribute_id (str) – ID or UUID

Return type

None

delete_object(*object_id*)

Delete an object

Parameters

object_id (str) – ID or UUID

Return type

None

property edited: **bool**

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

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

get_attribute_by_id(*attribute_id*)

Get an attribute by ID

Parameters

attribute_id (str | int) – The ID of the seeking attribute

Return type

MISPAttribute

get_attribute_by_uuid(*attribute_uuid*)

Get an attribute by UUID

Parameters

attribute_uuid (str) – The UUID of the seeking attribute

Return type

MISPAttribute

get_attribute_tag(*attribute_identifier*)

Return the tags associated to an attribute or an object attribute.

Parameters

attribute_identifier (str) – can be an ID, UUID, or the value.

Return type

list[*MISPTag*]

get_object_by_id(*object_id*)

Get an object by ID

Parameters

object_id (str | int) – the ID is the one set by the server when creating the new object

Return type

MISPObject

get_object_by_uuid(*object_uuid*)

Get an object by UUID

Parameters

object_uuid (str) – the UUID is set by the server when creating the new object

Return type

MISPObject

get_objects_by_name(*object_name*)

Get objects by name

Parameters

object_name (str) – name is set by the server when creating the new object

Return type

list[*MISPObject*]

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

jsonable()

This method is used by the JSON encoder

Return type

dict[str, Any]

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

load(*json_event*, *validate=False*, *metadata_only=False*)

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

Return type

None

load_file(*event_path*, *validate=False*, *metadata_only=False*)

Load a JSON dump from a file on the disk

Return type

None

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.

publish()

Mark the attribute as published

Return type

None

set_date(*d=None*, *ignore_invalid=False*)

Set a date for the event

Parameters

- **d** (`str` | `int` | `float` | `datetime` | `date` | `None`) – String, datetime, or date object
- **ignore_invalid** (`bool`) – if `True`, assigns current date if *d* is not an expected type

Return type

None

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

property tags: `list[MISPTag]`

Returns a list of tags associated to this Event

to_dict(*json_format=False*)

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.

Return type

`dict[str, Any]`

to_feed(*valid_distributions=[0, 1, 2, 3, 4, 5]*, *with_meta=False*, *with_distribution=False*,
with_local_tags=True, *with_event_reports=True*, *with_cryptographic_keys=True*)

Generate a json output for MISP Feed.

Parameters

- **valid_distributions** (`list[int]`) – only makes sense if the distribution key is set; i.e., the event is exported from a MISP instance.
- **with_distribution** (`bool`) – exports distribution and Sharing Group info; otherwise all SharingGroup information is discarded (protecting privacy)
- **with_local_tags** (`bool`) – tag export includes local exportable tags along with global exportable tags

- **with_event_reports** (bool) – include event reports in the returned MISP event
- **with_cryptographic_keys** (bool) – include the associated cryptographic keys in the returned protected MISP event

Return type
dict[str, Any]

to_json(*sort_keys=False, indent=None*)

Dump recursively any class of type MISPAbstract to a json string

Return type
str

unpublish()

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

Return type
None

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.keys(): 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

Return type
None

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

3.5 MISPEventBlocklist

class pymisp.MISPEventBlocklist(*force_timestamps=False, **kwargs*)

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

property edited: bool

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.

Return type
None

from_json(*json_string*)

Load a JSON string

Return type
None

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

Return type

dict[str, Any]

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.

set_not_jsonable(args)

Set __not_jsonable to a new list

Return type

None

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

to_dict(json_format=False)

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.

Return type

dict[str, Any]

to_json(sort_keys=False, indent=None)

Dump recursively any class of type MISPAbstract to a json string

Return type

str

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.keys(): 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

Return type

None

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

3.6 MISPEventDelegation

class pymisp.MISPEventDelegation(force_timestamps=False, **kwargs)

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

property edited: bool

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False*, *indent=None*)

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

Return type

str

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*.keys(): *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

Return type

None

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

3.7 MISPAtribute

class pymisp.MISPAtribute(*describe_types=None, strict=False*)

add_galaxy(*galaxy=None, **kwargs*)

Add a galaxy to the Attribute, either by passing a MISPGalaxy or a dictionary

Return type

MISPGalaxy

add_proposal(*shadow_attribute=None, **kwargs*)

Alias for add_shadow_attribute

Return type

MISPShadowAttribute

add_shadow_attribute(*shadow_attribute=None, **kwargs*)

Add a shadow attribute to the attribute (by name or a MISPSHadowAttribute object)

Return type

MISPShadowAttribute

add_sighting(*sighting=None, **kwargs*)

Add a sighting to the attribute (by name or a MISPSighting object)

Return type

MISPSighting

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

delete()

Mark the attribute as deleted (soft delete)

Return type

None

property edited: **bool**

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

property galaxies: `list[MISPGalaxy]`

Returns a list of galaxies associated to this Attribute

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

hash_values(*algorithm='sha512'*)

Compute the hash of every value for fast lookups

Return type

`list[str]`

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

jsonable()

This method is used by the JSON encoder

Return type

`dict[str, Any]`

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

property known_types: `list[str]`

Returns a list of all the known MISP attributes types

property malware_binary: `BytesIO | None`

Returns a BytesIO of the malware, if the attribute has one. Decrypts, unpacks and caches the binary on the first invocation, which may require some time for large attachments (~1s/MB).

`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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

`None`

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

property tags: `list[MISPTag]`

Returns a list of tags associated to this Attribute

to_dict(*json_format=False*)

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.

Return type

`dict[str, Any]`

to_json(*sort_keys=False, indent=None*)

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

Return type

`str`

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.keys(): 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

Return type

None

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

3.8 MISPObject

class pymisp.MISPObject(*name, strict=False, standalone=True, default_attributes_parameters={}, force_timestamps=False, **kwargs*)

add_attribute(*object_relation, simple_value=None, **value*)

Add an attribute. :type object_relation: str :param object_relation: The object relation of the attribute you're adding to the object :type simple_value: str | int | float | None :param simple_value: The value :type value: :param value: dictionary with all the keys supported by MISPAtribute

Return type

MISPAtribute | None

Note: as long as PyMISP knows about the object template, only the object_relation and the simple_value are required.

If PyMISP doesn't know the template, you also **must** pass a type. All the other options that can be passed along when creating an attribute (comment, IDS flag, ...) will be either taked out of the template, or out of the default setting for the type as defined on the MISP instance.

add_attributes(*object_relation, *attributes*)

Add multiple attributes with the same object_relation. Helper for object_relation when multiple is True in the template. It is the same as calling multiple times add_attribute with the same object_relation.

Return type

list[*MISPAtribute* | None]

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

Add a link (uuid) to another object

Return type

MISPObjectReference

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

delete()

Mark the object as deleted (soft delete)

Return type

None

property edited: bool

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

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

get_attribute_by_id(*attribute_id*)

Get an object attribute by ID

Parameters

attribute_id (str | int) – The ID of the seeking object attribute

Return type

MISPObjectAttribute

get_attribute_by_uuid(*attribute_uuid*)

Get an object attribute by UUID

Parameters

attribute_uuid (str) – The UUID of the seeking object attribute

Return type

MISPObjectAttribute

get_attributes_by_relation(*object_relation*)

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

Return type

list[*MISPAttribute*]

has_attributes_by_relation(*list_of_relations*)

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

Return type

bool

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

jsonable()

This method is used by the JSON encoder

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False*, *strict=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False*, *indent=None*, *strict=False*)

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

Return type

str

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*.keys(): *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

Return type

None

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

3.9 MISPOBJECTATTRIBUTE

class `pymisp.MISPOBJECTATTRIBUTE`(*definition*, *force_timestamps=False*)

add_galaxy(*galaxy=None*, ***kwargs*)

Add a galaxy to the Attribute, either by passing a `MISPGalaxy` or a dictionary

Return type

`MISPGalaxy`

add_proposal(*shadow_attribute=None*, ***kwargs*)

Alias for `add_shadow_attribute`

Return type

`MISPShadowAttribute`

add_shadow_attribute(*shadow_attribute=None*, ***kwargs*)

Add a shadow attribute to the attribute (by name or a `MISPShadowAttribute` object)

Return type

`MISPShadowAttribute`

add_sighting(*sighting=None*, ***kwargs*)

Add a sighting to the attribute (by name or a `MISPSighting` object)

Return type

`MISPSighting`

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

delete()

Mark the attribute as deleted (soft delete)

Return type

None

property edited: bool

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

from_dict(*object_relation*="", *value*="", ***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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

property galaxies: list[MISPGalaxy]

Returns a list of galaxies associated to this Attribute

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

hash_values(*algorithm*='sha512')

Compute the hash of every value for fast lookups

Return type

list[str]

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

jsonable()

This method is used by the JSON encoder

Return type

dict[str, Any]

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

property known_types: list[str]

Returns a list of all the known MISP attributes types

property malware_binary: BytesIO | None

Returns a BytesIO of the malware, if the attribute has one. Decrypts, unpacks and caches the binary on the first invocation, which may require some time for large attachments (~1s/MB).

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

property tags: list[MISPTag]

Returns a list of tags associated to this Attribute

to_dict(*json_format=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False*, *indent=None*)

Dump recursively any class of type MISPAbstract to a json string

Return type

str

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*.keys(): *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

Return type

None

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

3.10 MISPObjctReference

class pymisp.MISPObjctReference

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

property edited: bool

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

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

Return type
dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type
None

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

to_dict(*json_format=False*)

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.

Return type
dict[str, Any]

to_json(*sort_keys=False*, *indent=None*)

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

Return type
str

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*.keys(): 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

Return type
None

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

3.11 MISPObjecTemplate

class pymisp.MISPObjecTemplate(*force_timestamps=False*)

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

property edited: `bool`

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.

Return type

`None`

from_json(*json_string*)

Load a JSON string

Return type

`None`

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

Return type

`dict[str, Any]`

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

`None`

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

to_dict(*json_format=False*)

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.

Return type

`dict[str, Any]`

to_json(*sort_keys=False*, *indent=None*)

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

Return type

`str`

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.keys()*: `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

Return type

None

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

3.12 MISPTag

class pymisp.MISPTag(*force_timestamps=False, **kwargs*)

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

property edited: **bool**

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.

Return type

None

from_json(json_string)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(args)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False, indent=None*)

Dump recursively any class of type MISPAbstract to a json string

Return type

str

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.keys(): 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

Return type

None

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

3.13 MISPUser

class pymisp.MISPUser(*force_timestamps=False, **kwargs*)

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

property edited: bool

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False*)

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.

Return type

`dict[str, Any]`

to_json(*sort_keys=False*, *indent=None*)

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

Return type

`str`

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.keys()`: `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

Return type

None

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

3.14 MISPUserSetting

class `pymisp.MISPUserSetting(force_timestamps=False, **kwargs)`

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

property edited: `bool`

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False*, *indent=None*)

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

Return type

str

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*.keys(): *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

Return type

None

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

3.15 MISPOrganisation

class pymisp.MISPOrganisation

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

property edited: **bool**

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.

Return type

None

from_json(json_string)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(args)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(json_format=False)

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.

Return type

dict[str, Any]

to_json(sort_keys=False, indent=None)

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

Return type

str

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.keys(): 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

Return type

None

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

3.16 MISPOrganisationBlocklist

class pymisp.MISPOrganisationBlocklist(*force_timestamps=False*, ***kwargs*)**clear**() → None. Remove all items from D.**property edited:** bool

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False*, *indent=None*)

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

Return type

str

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*.keys(): *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

Return type

None

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

3.17 MISPFeeD

class `pymisp.MISPFeeD`(*force_timestamps=False*)

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

property edited: bool

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(args)

Set __not_jsonable to a new list

Return type

None

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

to_dict(json_format=False)

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.

Return type

dict[str, Any]

to_json(sort_keys=False, indent=None)

Dump recursively any class of type MISPAbstract to a json string

Return type

str

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.keys(): 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

Return type

None

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

3.18 MISPInbox

class pymisp.MISPInbox(force_timestamps=False, **kwargs)

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

property edited: bool

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False*, *indent=None*)

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

Return type

str

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*.keys(): *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

Return type

None

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

3.19 MISPLog

class `pymisp.MISPLog`(*force_timestamps=False*, ***kwargs*)

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

property edited: **bool**

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False, indent=None*)

Dump recursively any class of type MISPAbstract to a json string

Return type

str

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.keys(): 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

Return type

None

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

3.20 MISPNoticelist

class pymisp.MISPNoticelist(*force_timestamps=False*)

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

property edited: bool

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False*)

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.

Return type

`dict[str, Any]`

to_json(*sort_keys=False*, *indent=None*)

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

Return type

`str`

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.keys()*: *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

Return type

None

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

3.21 MISPRole

class `pymisp.MISPRole`(*force_timestamps=False*, ***kwargs*)

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

property edited: `bool`

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False*, *indent=None*)

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

Return type

str

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*.keys(): *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

Return type

None

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

3.22 MISPServer

`class pymisp.MISPServer(force_timestamps=False)`

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

property edited: `bool`

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.

Return type

None

from_json(json_string)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(args)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(json_format=False)

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.

Return type

dict[str, Any]

to_json(sort_keys=False, indent=None)

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

Return type

str

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.keys(): 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

Return type

None

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

3.23 MISPSHadowAttribute

class pymisp.MISPSHadowAttribute**clear**() → None. Remove all items from D.**property edited:** bool

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.

Return type

None

from_json(json_string)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False*, *indent=None*)

Dump recursively any class of type MISPAbstract to a json string

Return type

str

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*.keys(): *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

Return type

None

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

3.24 MISPSHaringGroup

class pymisp.MISPSHaringGroup

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

property edited: bool

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False*, *indent=None*)

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

Return type

str

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.keys()*: *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

Return type

None

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

3.25 MISPSighting

class pymisp.MISPSighting

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

property edited: bool

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

from_dict(***kwards*)

Initialize the MISPSighting from a dictionary

Parameters

- **value** – Value of the attribute the sighting is related too. Pushing this object will update the sighting count of each attribute with this value on the instance.
- **uuid** – UUID of the attribute to update
- **id** – ID of the attriute to update
- **source** – Source of the sighting
- **type** – Type of the sighting
- **timestamp** – Timestamp associated to the sighting

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set __not_jsonable to a new list

Return type

None

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

to_dict(*json_format=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False, indent=None*)

Dump recursively any class of type MISPAbstract to a json string

Return type

str

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.keys(): 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

Return type

None

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

3.26 MISPTaxonomy

class pymisp.MISPTaxonomy(*force_timestamps=False*)

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

property edited: bool

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False*, *indent=None*)

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

Return type

str

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*.keys(): *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

Return type

None

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

3.27 MISPPWarninglist

class `pymisp.MISPPWarninglist`(*force_timestamps=False*)

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

property edited: bool

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

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

Return type

dict[str, Any]

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.

set_not_jsonable(args)

Set __not_jsonable to a new list

Return type

None

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

to_dict(json_format=False)

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.

Return type

dict[str, Any]

to_json(sort_keys=False, indent=None)

Dump recursively any class of type MISPAbstract to a json string

Return type

str

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.keys(): 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

Return type

None

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

PYMISP - TOOLS

```
class pymisp.tools.ASNObject(parameters, strict=True, **kwargs)

    generate_attributes()
        Contains the logic where all the values of the object are gathered

        Return type
            None

class pymisp.tools.AbstractMISPObjectGenerator(name, strict=False, standalone=True,
                                                default_attributes_parameters={},
                                                force_timestamps=False, **kwargs)

    generate_attributes()
        Contains the logic where all the values of the object are gathered

        Return type
            None

class pymisp.tools.DataURLObject(dataurl, **kwargs)

    generate_attributes()
        Contains the logic where all the values of the object are gathered

        Return type
            None

class pymisp.tools.DomainIPObject(parameters, strict=True, **kwargs)

    generate_attributes()
        Contains the logic where all the values of the object are gathered

        Return type
            None

class pymisp.tools.ELFObject(parsed=None, filepath=None, pseudofile=None, **kwargs)

    generate_attributes()
        Contains the logic where all the values of the object are gathered

        Return type
            None

class pymisp.tools.ELFSectionObject(section, **kwargs)
```

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

```
class pymisp.tools.Fail2BanObject(parameters, strict=True, **kwargs)
```

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

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

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

```
class pymisp.tools.GenericObjectGenerator(name, strict=False, standalone=True,
                                          default_attributes_parameters={}, force_timestamps=False,
                                          **kwargs)
```

generate_attributes(attributes)

Generates MISPObjAttributes from a list of dictionaries. Each entry if the list must be in one of the two following formats: * {<object_relation>: <value>} * {<object_relation>: {'value'=<value>, 'type'=<type>, <and any other key/value accepted by a MISPObjAttribute>}}

Return type

None

Note: Any missing parameter will default to the pre-defined value from the Object template.

If the object template isn't known by PyMISP, you *must* pass a type key/value, or it will fail.

Example:

```
[{'analysis_submitted_at': '2018-06-15T06:40:27'}, {'threat_score': {'value=95, to_ids=False'}},
 {'permalink': 'https://panacea.threatgrid.com/mask/samples/2e445ef5389d8b'}, {'heuristic_raw_score': 7.8385159793597}, {'heuristic_score': 96}, {'original_filename': 'juice.exe'},
 {'id': '2e445ef5389d8b'}]
```

```
class pymisp.tools.GeolocationObject(parameters, strict=True, **kwargs)
```

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

```
class pymisp.tools.GitVulnFinderObject(parameters, strict=True, **kwargs)
```

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

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

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

class pymisp.tools.MachOSectionObject(*section, **kwargs*)

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

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

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

class pymisp.tools.PESectionObject(*section, **kwargs*)

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

class pymisp.tools.SBSignatureObject(*software, report, **kwargs*)

Sandbox Analyzer

generate_attributes()

Parse the report for relevant attributes

Return type

None

class pymisp.tools.SSHAuthorizedKeysObject(*authorized_keys_path=None, authorized_keys_pseudofile=None, **kwargs*)

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

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

VirusTotal Report

Apikey

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

Indicator

IOC to search VirusTotal for

generate_attributes()

Parse the VirusTotal report for relevant attributes

Return type

None

exception `pymisp.tools.ValidationError`(*message*)

class `pymisp.tools.VehicleObject`(*country, registration, username, **kwargs*)

Vehicle object generator out of regcheck.org.uk

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

`pymisp.tools.validate_attribute`(*attribute*)

Validates a MISP Attribute and returns a MISPAttribute if valid. Replicates MISP server-side validation behavior on Attributes.

Parameters

attribute (dict | *MISPAttribute*) – dict or MISPAttribute to validate

Return type

MISPAttribute

Returns

Validated MISPAttribute object

Raises

- *PyMISPError* – If the attribute cannot be loaded or a validation error occurs
- *ValidationError* – If the attribute is invalid

`pymisp.tools.validate_attributes`(*attributes, errors*)

Validates a list of MISP attributes and skips any that doesn't validate.

Parameters

- **attributes** (list) – List of MISPAttribute objects
- **errors** (dict) – Dictionary to populate with any validation error messages

Return type

Generator

Returns

Generator yielding only valid MISPAttribute objects

`pymisp.tools.validate_event`(*event, errors*)

Validates an event and skips Attributes or Object Attributes that don't validate.

Parameters

- **event** (dict | *MISPEvent*) – MISPEvent object or dict representing an event
- **errors** (dict) – Dictionary to populate with any validation error messages

Return type

MISPEvent

Returns

MISPEvent with only valid attributes

Raises

PyMISPError – If the event cannot be loaded

`pymisp.tools.validate_object(misp_object, errors)`

Validates an object and skips any Object Attribute that doesn't validate.

Parameters

- **misp_object** (dict | *MISPObject*) – MISPObject object or dict representing an object
- **errors** (dict) – Dictionary to populate with any validation error messages

Return type

MISPObject

Returns

MISPObject with only valid attributes

Raises

PyMISPError – If the object cannot be loaded

`pymisp.tools.validate_objects(misp_objects, errors)`

Validates a list of MISP objects and skips any Object Attribute that doesn't validate.

Parameters

- **misp_objects** (list) – List of MISPObject objects
- **errors** (dict) – Dictionary to populate with any validation error messages

Return type

Generator

Returns

Generator yielding only valid MISPObject objects

4.1 Excel / CSV Importer

If the header of the CSV file has valid object relations in the template you're using:

```
from pymisp.tools import CSVLoader
from pymisp import MISPEvent
from pathlib import Path

csv1 = CSVLoader(template_name='file', csv_path=Path('tests/csv_testfiles/valid_
↪fieldnames.csv'))
event = MISPEvent()
event.info = 'Test event from CSV loader'
for o in csv1.load():
    event.add_object(**o)
```

If the header of the CSV file does not have valid object relations in the template you're using:

```
event = MISPEvent()
event.info = 'Test event from CSV loader'
csv2 = CSVLoader(template_name='file', csv_path=Path('tests/csv_testfiles/invalid_
```

(continues on next page)

(continued from previous page)

```

↪fieldnames.csv'),
        fieldnames=['SHA1', 'fileName', 'size-in-bytes'], has_fieldnames=True)

for o in csv2.load():
    event.add_object(**o)

```

```

class pymisp.tools.CSVLoader(template_name, csv_path, fieldnames=None, has_fieldnames=False,
                             delimiter=',', quotechar='')

```

4.2 File Object

```

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

```

```

add_attribute(object_relation, simple_value=None, **value)

```

Add an attribute. :type object_relation: str :param object_relation: The object relation of the attribute you're adding to the object :type simple_value: str | int | float | None :param simple_value: The value :type value: :param value: dictionary with all the keys supported by MISPAtribute

Return type

MISPAtribute | None

Note: as long as PyMISP knows about the object template, only the `object_relation` and the `simple_value` are required.

If PyMISP doesn't know the template, you also **must** pass a type. All the other options that can be passed along when creating an attribute (comment, IDS flag, ...) will be either taken out of the template, or out of the default setting for the type as defined on the MISP instance.

```

add_attributes(object_relation, *attributes)

```

Add multiple attributes with the same `object_relation`. Helper for `object_relation` when `multiple` is True in the template. It is the same as calling multiple times `add_attribute` with the same `object_relation`.

Return type

list[*MISPAtribute* | None]

```

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

```

Add a link (uuid) to another object

Return type

MISPObjectReference

```

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

```

```

delete()

```

Mark the object as deleted (soft delete)

Return type

None

```

property edited: bool

```

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

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

Get an object attribute by ID

Parameters**attribute_id** (str | int) – The ID of the seeking object attribute**Return type***MISPObjectAttribute***get_attribute_by_uuid**(*attribute_uuid*)

Get an object attribute by UUID

Parameters**attribute_uuid** (str) – The UUID of the seeking object attribute**Return type***MISPObjectAttribute***get_attributes_by_relation**(*object_relation*)

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

Return typelist[*MISPAttribute*]**has_attributes_by_relation**(*list_of_relations*)

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

Return type

bool

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

This method is used by the JSON encoder

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)
Set `__not_jsonable` to a new list

Return type
None

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

to_dict(*json_format=False, strict=False*)

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.

Return type
dict[str, Any]

to_json(*sort_keys=False, indent=None, strict=False*)

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

Return type
str

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.keys(): 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

Return type
None

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

4.3 ELF Object

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

add_attribute(*object_relation, simple_value=None, **value*)

Add an attribute. :type object_relation: str :param object_relation: The object relation of the attribute you're adding to the object :type simple_value: str | int | float | None :param simple_value: The value :type value: :param value: dictionary with all the keys supported by `MISPAttribute`

Return type
`MISPAttribute` | None

Note: as long as PyMISP knows about the object template, only the `object_relation` and the `simple_value` are required.

If PyMISP doesn't know the template, you also **must** pass a type. All the other options that can be passed along when creating an attribute (comment, IDS flag, ...) will be either taken out of the template, or out of the default setting for the type as defined on the MISP instance.

add_attributes(*object_relation, *attributes*)

Add multiple attributes with the same `object_relation`. Helper for `object_relation` when `multiple` is True in the template. It is the same as calling multiple times `add_attribute` with the same `object_relation`.

Return type

list[MISPAttribute | None]

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

Add a link (uuid) to another object

Return type

MISPObjectReference

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

Mark the object as deleted (soft delete)

Return type

None

property edited: bool

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.

Return type

None

from_json(json_string)

Load a JSON string

Return type

None

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

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

Get an object attribute by ID

Parameters**attribute_id** (str | int) – The ID of the seeking object attribute**Return type**

MISPObjectAttribute

get_attribute_by_uuid(attribute_uuid)

Get an object attribute by UUID

Parameters**attribute_uuid** (str) – The UUID of the seeking object attribute**Return type**

MISPObjectAttribute

get_attributes_by_relation(*object_relation*)

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

Return type

list[MISPAttribute]

has_attributes_by_relation(*list_of_relations*)

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

Return type

bool

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

jsonable()

This method is used by the JSON encoder

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False, strict=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False, indent=None, strict=False*)

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

Return type

str

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.keys()*: *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

Return type

None

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

```
class pymisp.tools.ELFSectionObject(section, **kwargs)
```

```
add_attribute(object_relation, simple_value=None, **value)
```

Add an attribute. :type object_relation: str :param object_relation: The object relation of the attribute you're adding to the object :type simple_value: str | int | float | None :param simple_value: The value :type value: :param value: dictionary with all the keys supported by MISPAtribute

Return type

MISPAtribute | None

Note: as long as PyMISP knows about the object template, only the object_relation and the simple_value are required.

If PyMISP doesn't know the template, you also **must** pass a type. All the other options that can be passed along when creating an attribute (comment, IDS flag, ...) will be either taked out of the template, or out of the default setting for the type as defined on the MISP instance.

```
add_attributes(object_relation, *attributes)
```

Add multiple attributes with the same object_relation. Helper for object_relation when multiple is True in the template. It is the same as calling multiple times add_attribute with the same object_relation.

Return type

list[*MISPAtribute* | None]

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

Add a link (uuid) to another object

Return type

MISPObjectReference

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

```
delete()
```

Mark the object as deleted (soft delete)

Return type

None

```
property edited: bool
```

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.

Return type

None

```
from_json(json_string)
```

Load a JSON string

Return type

None

```
generate_attributes()
```

Contains the logic where all the values of the object are gathered

Return type

None

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

get_attribute_by_id(*attribute_id*)

Get an object attribute by ID

Parameters

attribute_id (str | int) – The ID of the seeking object attribute

Return type

MISPObjectAttribute

get_attribute_by_uuid(*attribute_uuid*)

Get an object attribute by UUID

Parameters

attribute_uuid (str) – The UUID of the seeking object attribute

Return type

MISPObjectAttribute

get_attributes_by_relation(*object_relation*)

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

Return type

list[*MISPAttribute*]

has_attributes_by_relation(*list_of_relations*)

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

Return type

bool

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

jsonable()

This method is used by the JSON encoder

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False*, *strict=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False, indent=None, strict=False*)

Dump recursively any class of type MISPAbstract to a json string

Return type

str

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.keys(): 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

Return type

None

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

4.4 PE Object

class pymisp.tools.**PEObject**(*parsed=None, filepath=None, pseudofile=None, **kwargs*)**add_attribute**(*object_relation, simple_value=None, **value*)

Add an attribute. :type object_relation: str :param object_relation: The object relation of the attribute you're adding to the object :type simple_value: str | int | float | None :param simple_value: The value :type value: :param value: dictionary with all the keys supported by MISPAtribute

Return type*MISPAtribute* | None**Note:** as long as PyMISP knows about the object template, only the object_relation and the simple_value are required.If PyMISP doesn't know the template, you also **must** pass a type. All the other options that can be passed along when creating an attribute (comment, IDS flag, ...) will be either taked out of the template, or out of the default setting for the type as defined on the MISP instance.**add_attributes**(*object_relation, *attributes*)

Add multiple attributes with the same object_relation. Helper for object_relation when multiple is True in the template. It is the same as calling multiple times add_attribute with the same object_relation.

Return typelist[*MISPAtribute* | None]**add_reference**(*referenced_uuid, relationship_type, comment=None, **kwargs*)

Add a link (uuid) to another object

Return type*MISPObjectReference***clear**() → None. Remove all items from D.**delete**()

Mark the object as deleted (soft delete)

Return type

None

property edited: bool

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.

Return type

None

from_json(json_string)

Load a JSON string

Return type

None

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

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

get_attribute_by_id(attribute_id)

Get an object attribute by ID

Parameters

attribute_id (str | int) – The ID of the seeking object attribute

Return type*MISPObjectAttribute***get_attribute_by_uuid(attribute_uuid)**

Get an object attribute by UUID

Parameters

attribute_uuid (str) – The UUID of the seeking object attribute

Return type*MISPObjectAttribute***get_attributes_by_relation(object_relation)**

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

Return typelist[*MISPAttribute*]**has_attributes_by_relation(list_of_relations)**

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

Return type

bool

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

jsonable()

This method is used by the JSON encoder

Return type

dict[str, Any]

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.

set_not_jsonable(args)

Set __not_jsonable to a new list

Return type

None

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

to_dict(json_format=False, strict=False)

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.

Return type

dict[str, Any]

to_json(sort_keys=False, indent=None, strict=False)

Dump recursively any class of type MISPAbstract to a json string

Return type

str

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.keys(): 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

Return type

None

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

class pymisp.tools.PESectionObject(section, **kwargs)

add_attribute(object_relation, simple_value=None, **value)

Add an attribute. :type object_relation: str :param object_relation: The object relation of the attribute you're adding to the object :type simple_value: str | int | float | None :param simple_value: The value :type value: :param value: dictionary with all the keys supported by MISPAttribute

Return type

MISPAttribute | None

Note: as long as PyMISP knows about the object template, only the object_relation and the simple_value are required.

If PyMISP doesn't know the template, you also **must** pass a type. All the other options that can

be passed along when creating an attribute (comment, IDS flag, ...) will be either taken out of the template, or out of the default setting for the type as defined on the MISP instance.

add_attributes(*object_relation*, **attributes*)

Add multiple attributes with the same *object_relation*. Helper for *object_relation* when multiple is True in the template. It is the same as calling multiple times *add_attribute* with the same *object_relation*.

Return type

`list[MISPAttribute | None]`

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

Add a link (uuid) to another object

Return type

`MISPObjectReference`

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

delete()

Mark the object as deleted (soft delete)

Return type

None

property edited: bool

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

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

get_attribute_by_id(*attribute_id*)

Get an object attribute by ID

Parameters

attribute_id (str | int) – The ID of the seeking object attribute

Return type

`MISPObjectAttribute`

get_attribute_by_uuid(*attribute_uuid*)

Get an object attribute by UUID

Parameters

attribute_uuid (*str*) – The UUID of the seeking object attribute

Return type

MISPObjectAttribute

get_attributes_by_relation(*object_relation*)

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

Return type

list[*MISPAttribute*]

has_attributes_by_relation(*list_of_relations*)

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

Return type

bool

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

jsonable()

This method is used by the JSON encoder

Return type

dict[*str*, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False*, *strict=False*)

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.

Return type

dict[*str*, Any]

to_json(*sort_keys=False*, *indent=None*, *strict=False*)

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

Return type

str

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.keys()*: *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

Return type

None

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

4.5 Mach-O Object

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

`add_attribute(object_relation, simple_value=None, **value)`

Add an attribute. :type object_relation: str :param object_relation: The object relation of the attribute you're adding to the object :type simple_value: str | int | float | None :param simple_value: The value :type value: :param value: dictionary with all the keys supported by MISPAtribute

Return type

`MISPAtribute` | None

Note: as long as PyMISP knows about the object template, only the `object_relation` and the `simple_value` are required.

If PyMISP doesn't know the template, you also **must** pass a type. All the other options that can be passed along when creating an attribute (comment, IDS flag, ...) will be either taken out of the template, or out of the default setting for the type as defined on the MISP instance.

`add_attributes(object_relation, *attributes)`

Add multiple attributes with the same `object_relation`. Helper for `object_relation` when `multiple` is True in the template. It is the same as calling multiple times `add_attribute` with the same `object_relation`.

Return type

`list[MISPAtribute]` | None

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

Add a link (uuid) to another object

Return type

`MISPObjectReference`

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

`delete()`

Mark the object as deleted (soft delete)

Return type

None

property edited: bool

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.

Return type

None

from_json(*json_string*)

Load a JSON string

Return type

None

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

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

get_attribute_by_id(*attribute_id*)

Get an object attribute by ID

Parameters

attribute_id (str | int) – The ID of the seeking object attribute

Return type

MISPObjectAttribute

get_attribute_by_uuid(*attribute_uuid*)

Get an object attribute by UUID

Parameters

attribute_uuid (str) – The UUID of the seeking object attribute

Return type

MISPObjectAttribute

get_attributes_by_relation(*object_relation*)

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

Return type

list[*MISPAttribute*]

has_attributes_by_relation(*list_of_relations*)

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

Return type

bool

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

jsonable()

This method is used by the JSON encoder

Return type

dict[str, Any]

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.

set_not_jsonable(args)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(json_format=False, strict=False)

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.

Return type

dict[str, Any]

to_json(sort_keys=False, indent=None, strict=False)

Dump recursively any class of type MISPAbstract to a json string

Return type

str

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.keys(): 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

Return type

None

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

class pymisp.tools.MachOSectionObject(section, **kwargs)

add_attribute(object_relation, simple_value=None, **value)

Add an attribute. :type object_relation: str :param object_relation: The object relation of the attribute you're adding to the object :type simple_value: str | int | float | None :param simple_value: The value :type value: :param value: dictionary with all the keys supported by MISPAttribute

Return type

MISPAttribute | None

Note: as long as PyMISP knows about the object template, only the object_relation and the simple_value are required.

If PyMISP doesn't know the template, you also **must** pass a type. All the other options that can be passed along when creating an attribute (comment, IDS flag, ...) will be either taked out of the template, or out of the default setting for the type as defined on the MISP instance.

add_attributes(object_relation, *attributes)

Add multiple attributes with the same object_relation. Helper for object_relation when multiple is True in the template. It is the same as calling multiple times add_attribute with the same object_relation.

Return type

list[MISPAttribute | None]

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

Add a link (uuid) to another object

Return type*MISPObjectReference***clear()** → None. Remove all items from D.**delete()**

Mark the object as deleted (soft delete)

Return type

None

property edited: bool

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.**Return type**

None

from_json(json_string)

Load a JSON string

Return type

None

generate_attributes()

Contains the logic where all the values of the object are gathered

Return type

None

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

Get an object attribute by ID

Parameters**attribute_id** (str | int) – The ID of the seeking object attribute**Return type***MISPObjectAttribute***get_attribute_by_uuid(attribute_uuid)**

Get an object attribute by UUID

Parameters**attribute_uuid** (str) – The UUID of the seeking object attribute**Return type***MISPObjectAttribute***get_attributes_by_relation(object_relation)**

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

Return typelist[*MISPAttribute*]

has_attributes_by_relation(*list_of_relations*)

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

Return type

bool

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

jsonable()

This method is used by the JSON encoder

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False*, *strict=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False*, *indent=None*, *strict=False*)

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

Return type

str

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.keys()*: *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

Return type

None

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

4.6 VT Report Object

`class pymisp.tools.VTReportObject(apikey, indicator, vt_proxies=None, **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, simple_value=None, **value)`

Add an attribute. :type object_relation: str :param object_relation: The object relation of the attribute you're adding to the object :type simple_value: str | int | float | None :param simple_value: The value :type value: :param value: dictionary with all the keys supported by MISPAtribute

Return type

`MISPAtribute` | None

Note: as long as PyMISP knows about the object template, only the object_relation and the simple_value are required.

If PyMISP doesn't know the template, you also **must** pass a type. All the other options that can be passed along when creating an attribute (comment, IDS flag, ...) will be either taked out of the template, or out of the default setting for the type as defined on the MISP instance.

`add_attributes(object_relation, *attributes)`

Add multiple attributes with the same object_relation. Helper for object_relation when multiple is True in the template. It is the same as calling multiple times add_attribute with the same object_relation.

Return type

`list[MISPAtribute` | None]

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

Add a link (uuid) to another object

Return type

`MISPObjectReference`

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

`delete()`

Mark the object as deleted (soft delete)

Return type

None

property edited: bool

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.

Return type

None

`from_json(json_string)`

Load a JSON string

Return type

None

generate_attributes()

Parse the VirusTotal report for relevant attributes

Return type

None

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

get_attribute_by_id(*attribute_id*)

Get an object attribute by ID

Parameters

attribute_id (str | int) – The ID of the seeking object attribute

Return type

MISPObjectAttribute

get_attribute_by_uuid(*attribute_uuid*)

Get an object attribute by UUID

Parameters

attribute_uuid (str) – The UUID of the seeking object attribute

Return type

MISPObjectAttribute

get_attributes_by_relation(*object_relation*)

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

Return type

list[*MISPAttribute*]

has_attributes_by_relation(*list_of_relations*)

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

Return type

bool

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

jsonable()

This method is used by the JSON encoder

Return type

dict[str, Any]

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.

set_not_jsonable(*args*)

Set `__not_jsonable` to a new list

Return type

None

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

to_dict(*json_format=False, strict=False*)

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.

Return type

dict[str, Any]

to_json(*sort_keys=False, indent=None, strict=False*)

Dump recursively any class of type MISPAbstract to a json string

Return type

str

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.keys(): 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

Return type

None

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

4.7 STIX

4.8 OpenIOC

tools.load_openioc()

tools.load_openioc_file()

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

`pymisp`, 9

`pymisp.tools`, 153

A

- AbstractMISP (class in *pymisp*), 9, 113
- AbstractMISPObjectGenerator (class in *pymisp.tools*), 153
- accept_attribute_proposal() (*pymisp.PyMISP* method), 23, 69
- accept_event_delegation() (*pymisp.PyMISP* method), 24, 69
- accept_user_registration() (*pymisp.PyMISP* method), 24, 69
- add_analyst_data() (*pymisp.PyMISP* method), 24, 69
- add_attribute() (*pymisp.MISPEvent* method), 11, 115
- add_attribute() (*pymisp.MISPObject* method), 18, 124
- add_attribute() (*pymisp.PyMISP* method), 24, 69
- add_attribute() (*pymisp.tools.ELFObject* method), 160
- add_attribute() (*pymisp.tools.ELFSectionObject* method), 163
- add_attribute() (*pymisp.tools.FileObject* method), 158
- add_attribute() (*pymisp.tools.MachOObject* method), 170
- add_attribute() (*pymisp.tools.MachOSectionObject* method), 172
- add_attribute() (*pymisp.tools.PEObject* method), 165
- add_attribute() (*pymisp.tools.PESectionObject* method), 167
- add_attribute() (*pymisp.tools.VTReportObject* method), 175
- add_attribute_proposal() (*pymisp.PyMISP* method), 25, 70
- add_attribute_tag() (*pymisp.MISPEvent* method), 11, 115
- add_attributes() (*pymisp.MISPObject* method), 18, 124
- add_attributes() (*pymisp.tools.ELFObject* method), 160
- add_attributes() (*pymisp.tools.ELFSectionObject* method), 163
- add_attributes() (*pymisp.tools.FileObject* method), 158
- add_attributes() (*pymisp.tools.MachOObject* method), 170
- add_attributes() (*pymisp.tools.MachOSectionObject* method), 172
- add_attributes() (*pymisp.tools.PEObject* method), 165
- add_attributes() (*pymisp.tools.PESectionObject* method), 168
- add_attributes() (*pymisp.tools.VTReportObject* method), 175
- add_cluster_element() (*pymisp.MISPGalaxyCluster* method), 16
- add_cluster_relation() (*pymisp.MISPGalaxyCluster* method), 16
- add_correlation_exclusion() (*pymisp.PyMISP* method), 25, 70
- add_cryprographic_key() (*pymisp.MISPEvent* method), 12, 115
- add_event() (*pymisp.PyMISP* method), 25, 70
- add_event_blocklist() (*pymisp.PyMISP* method), 25, 70
- add_event_report() (*pymisp.MISPEvent* method), 12, 115
- add_event_report() (*pymisp.PyMISP* method), 25, 71
- add_feed() (*pymisp.PyMISP* method), 26, 71
- add_galaxy() (*pymisp.MISPAtribute* method), 10, 122
- add_galaxy() (*pymisp.MISPEvent* method), 12, 115
- add_galaxy() (*pymisp.MISPObjectAttribute* method), 126
- add_galaxy_cluster() (*pymisp.MISPGalaxy* method), 15
- add_galaxy_cluster() (*pymisp.PyMISP* method), 26, 71
- add_galaxy_cluster_relation() (*pymisp.PyMISP* method), 26, 71
- add_note() (*pymisp.PyMISP* method), 26, 71
- add_object() (*pymisp.MISPEvent* method), 12, 115
- add_object() (*pymisp.PyMISP* method), 26, 72
- add_object_reference() (*pymisp.PyMISP* method), 27, 72
- add_opinion() (*pymisp.PyMISP* method), 27, 72
- add_org_to_sharing_group() (*pymisp.PyMISP*

method), 27, 72

add_organisation() (*pymisp.PyMISP method*), 27, 72

add_organisation_blocklist() (*pymisp.PyMISP method*), 27, 73

add_proposal() (*pymisp.MISPAttribute method*), 10, 122

add_proposal() (*pymisp.MISPEvent method*), 12, 115

add_proposal() (*pymisp.MISPObjectAttribute method*), 126

add_reference() (*pymisp.MISPObject method*), 19, 124

add_reference() (*pymisp.tools.ELFObject method*), 161

add_reference() (*pymisp.tools.ELFSectionObject method*), 163

add_reference() (*pymisp.tools.FileObject method*), 158

add_reference() (*pymisp.tools.MachOObject method*), 170

add_reference() (*pymisp.tools.MachOSectionObject method*), 172

add_reference() (*pymisp.tools.PEObject method*), 165

add_reference() (*pymisp.tools.PESectionObject method*), 168

add_reference() (*pymisp.tools.VTReportObject method*), 175

add_relationship() (*pymisp.PyMISP method*), 28, 73

add_role() (*pymisp.PyMISP method*), 28, 73

add_server() (*pymisp.PyMISP method*), 28, 73

add_server_to_sharing_group() (*pymisp.PyMISP method*), 28, 73

add_shadow_attribute() (*pymisp.MISPAttribute method*), 10, 122

add_shadow_attribute() (*pymisp.MISPEvent method*), 12, 116

add_shadow_attribute() (*pymisp.MISPObjectAttribute method*), 126

add_sharing_group() (*pymisp.PyMISP method*), 29, 74

add_sighting() (*pymisp.MISPAttribute method*), 10, 122

add_sighting() (*pymisp.MISPObjectAttribute method*), 126

add_sighting() (*pymisp.PyMISP method*), 29, 74

add_tag() (*pymisp.PyMISP method*), 29, 74

add_user() (*pymisp.PyMISP method*), 29, 74

Analysis (class in *pymisp*), 10

ASNObject (class in *pymisp.tools*), 153

attach_galaxy_cluster() (*pymisp.PyMISP method*), 29, 75

attribute_exists() (*pymisp.PyMISP method*), 30, 75

attribute_proposals() (*pymisp.PyMISP method*), 30, 75

attributes() (*pymisp.PyMISP method*), 30, 75

attributes_statistics() (*pymisp.PyMISP method*), 30, 75

B

build_complex_query() (*pymisp.PyMISP method*), 30, 76

C

cache_all_feeds() (*pymisp.PyMISP method*), 30, 76

cache_feed() (*pymisp.PyMISP method*), 31, 76

cache_freertext_feeds() (*pymisp.PyMISP method*), 31, 76

cache_misp_feeds() (*pymisp.PyMISP method*), 31, 76

change_sharing_group_on_entity() (*pymisp.PyMISP method*), 31, 76

change_user_password() (*pymisp.PyMISP method*), 31, 76

clean_correlation_exclusions() (*pymisp.PyMISP method*), 31, 76

clear() (*pymisp.MISPAttribute method*), 122

clear() (*pymisp.MISPEvent method*), 116

clear() (*pymisp.MISPEventBlocklist method*), 119

clear() (*pymisp.MISPEventDelegation method*), 120

clear() (*pymisp.MISPFeed method*), 137

clear() (*pymisp.MISPInbox method*), 138

clear() (*pymisp.MISPLog method*), 140

clear() (*pymisp.MISPNoticelist method*), 141

clear() (*pymisp.MISPObject method*), 124

clear() (*pymisp.MISPObjectAttribute method*), 127

clear() (*pymisp.MISPObjectReference method*), 128

clear() (*pymisp.MISPObjectTemplate method*), 129

clear() (*pymisp.MISPOrganisation method*), 135

clear() (*pymisp.MISPOrganisationBlocklist method*), 136

clear() (*pymisp.MISPRole method*), 142

clear() (*pymisp.MISPServer method*), 144

clear() (*pymisp.MISPShadowAttribute method*), 145

clear() (*pymisp.MISPSharingGroup method*), 146

clear() (*pymisp.MISP Sighting method*), 147

clear() (*pymisp.MISPTag method*), 131

clear() (*pymisp.MISPTaxonomy method*), 149

clear() (*pymisp.MISPUser method*), 132

clear() (*pymisp.MISPUserSetting method*), 133

clear() (*pymisp.MISPWarninglist method*), 150

clear() (*pymisp.tools.ELFObject method*), 161

clear() (*pymisp.tools.ELFSectionObject method*), 163

clear() (*pymisp.tools.FileObject method*), 158

clear() (*pymisp.tools.MachOObject method*), 170

clear() (*pymisp.tools.MachOSectionObject method*), 173

clear() (*pymisp.tools.PEObject method*), 165

clear() (*pymisp.tools.PESectionObject method*), 168

clear() (*pymisp.tools.VTReportObject method*), 175

communities() (*pymisp.PyMISP method*), 31, 77

compare_feeds() (*pymisp.PyMISP method*), 31, 77
 contact_event_reporter() (*pymisp.PyMISP method*), 32, 77
 correlation_exclusions() (*pymisp.PyMISP method*), 32, 77
 CSVLoader (*class in pymisp.tools*), 158

D

DataURLObject (*class in pymisp.tools*), 153
 db_schema_diagnostic() (*pymisp.PyMISP method*), 32, 77
 decaying_models() (*pymisp.PyMISP method*), 32, 77
 default() (*pymisp.MISPEncode method*), 114
 delegate_event() (*pymisp.PyMISP method*), 32, 77
 delete() (*pymisp.MISPAttribute method*), 10, 122
 delete() (*pymisp.MISPObject method*), 19, 124
 delete() (*pymisp.MISPObjectAttribute method*), 127
 delete() (*pymisp.tools.ELFObject method*), 161
 delete() (*pymisp.tools.ELFSectionObject method*), 163
 delete() (*pymisp.tools.FileObject method*), 158
 delete() (*pymisp.tools.MachOObject method*), 170
 delete() (*pymisp.tools.MachOSectionObject method*), 173
 delete() (*pymisp.tools.PEObject method*), 165
 delete() (*pymisp.tools.PESectionObject method*), 168
 delete() (*pymisp.tools.VTRReportObject method*), 175
 delete_analyst_data() (*pymisp.PyMISP method*), 32, 78
 delete_attribute() (*pymisp.MISPEvent method*), 12, 116
 delete_attribute() (*pymisp.PyMISP method*), 33, 78
 delete_attribute_proposal() (*pymisp.PyMISP method*), 33, 78
 delete_correlation_exclusion() (*pymisp.PyMISP method*), 33, 78
 delete_event() (*pymisp.PyMISP method*), 33, 78
 delete_event_blocklist() (*pymisp.PyMISP method*), 33, 79
 delete_event_report() (*pymisp.PyMISP method*), 33, 79
 delete_feed() (*pymisp.PyMISP method*), 34, 79
 delete_galaxy_cluster() (*pymisp.PyMISP method*), 34, 79
 delete_galaxy_cluster_relation() (*pymisp.PyMISP method*), 34, 79
 delete_note() (*pymisp.PyMISP method*), 34, 79
 delete_object() (*pymisp.MISPEvent method*), 12, 116
 delete_object() (*pymisp.PyMISP method*), 34, 80
 delete_object_reference() (*pymisp.PyMISP method*), 35, 80
 delete_opinion() (*pymisp.PyMISP method*), 35, 80
 delete_organisation() (*pymisp.PyMISP method*), 35, 80

delete_organisation_blocklist() (*pymisp.PyMISP method*), 35, 80
 delete_relationship() (*pymisp.PyMISP method*), 35, 80
 delete_role() (*pymisp.PyMISP method*), 35, 80
 delete_server() (*pymisp.PyMISP method*), 35, 81
 delete_sharing_group() (*pymisp.PyMISP method*), 36, 81
 delete_sighting() (*pymisp.PyMISP method*), 36, 81
 delete_tag() (*pymisp.PyMISP method*), 36, 81
 delete_user() (*pymisp.PyMISP method*), 36, 81
 delete_user_setting() (*pymisp.PyMISP method*), 36, 81
 describe_types_local (*pymisp.PyMISP property*), 36, 82
 describe_types_remote (*pymisp.PyMISP property*), 36, 82
 direct_call() (*pymisp.PyMISP method*), 36, 82
 disable_decaying_model() (*pymisp.PyMISP method*), 37, 82
 disable_feed() (*pymisp.PyMISP method*), 37, 82
 disable_feed_cache() (*pymisp.PyMISP method*), 37, 82
 disable_noticelist() (*pymisp.PyMISP method*), 37, 82
 disable_tag() (*pymisp.PyMISP method*), 37, 83
 disable_taxonomy() (*pymisp.PyMISP method*), 38, 83
 disable_taxonomy_tags() (*pymisp.PyMISP method*), 38, 83
 disable_warninglist() (*pymisp.PyMISP method*), 38, 83
 discard_attribute_proposal() (*pymisp.PyMISP method*), 38, 83
 discard_event_delegation() (*pymisp.PyMISP method*), 38, 83
 discard_user_registration() (*pymisp.PyMISP method*), 38, 83
 Distribution (*class in pymisp*), 10
 DomainIPObject (*class in pymisp.tools*), 153

E

edited (*pymisp.AbstractMISP property*), 9, 113
 edited (*pymisp.MISPAttribute property*), 122
 edited (*pymisp.MISPEvent property*), 116
 edited (*pymisp.MISPEventBlocklist property*), 119
 edited (*pymisp.MISPEventDelegation property*), 120
 edited (*pymisp.MISPFeed property*), 137
 edited (*pymisp.MISPInbox property*), 138
 edited (*pymisp.MISPLog property*), 140
 edited (*pymisp.MISPNoticelist property*), 141
 edited (*pymisp.MISPObject property*), 124
 edited (*pymisp.MISPObjectAttribute property*), 127
 edited (*pymisp.MISPObjectReference property*), 128
 edited (*pymisp.MISPObjectTemplate property*), 129

- edited (*pymisp.MISPOrganisation property*), 135
 edited (*pymisp.MISPOrganisationBlocklist property*), 136
 edited (*pymisp.MISPRole property*), 142
 edited (*pymisp.MISPServer property*), 144
 edited (*pymisp.MISPShadowAttribute property*), 145
 edited (*pymisp.MISPSharingGroup property*), 146
 edited (*pymisp.MISPSighting property*), 147
 edited (*pymisp.MISPTag property*), 131
 edited (*pymisp.MISPTaxonomy property*), 149
 edited (*pymisp.MISPUser property*), 132
 edited (*pymisp.MISPUserSetting property*), 133
 edited (*pymisp.MISPWarninglist property*), 150
 edited (*pymisp.tools.ELFObject property*), 161
 edited (*pymisp.tools.ELFSectionObject property*), 163
 edited (*pymisp.tools.FileObject property*), 158
 edited (*pymisp.tools.MachOObject property*), 170
 edited (*pymisp.tools.MachOSectionObject property*), 173
 edited (*pymisp.tools.PEObject property*), 166
 edited (*pymisp.tools.PESectionObject property*), 168
 edited (*pymisp.tools.VTRReportObject property*), 175
 elements_meta (*pymisp.MISPGalaxyCluster property*), 16
 ELFObject (*class in pymisp.tools*), 153, 160
 ELFSectionObject (*class in pymisp.tools*), 153, 162
 enable_decaying_model() (*pymisp.PyMISP method*), 38, 84
 enable_feed() (*pymisp.PyMISP method*), 38, 84
 enable_feed_cache() (*pymisp.PyMISP method*), 39, 84
 enable_noticelist() (*pymisp.PyMISP method*), 39, 84
 enable_tag() (*pymisp.PyMISP method*), 39, 84
 enable_taxonomy() (*pymisp.PyMISP method*), 39, 84
 enable_taxonomy_tags() (*pymisp.PyMISP method*), 39, 85
 enable_warninglist() (*pymisp.PyMISP method*), 39, 85
 enrich_attribute() (*pymisp.PyMISP method*), 40, 85
 enrich_event() (*pymisp.PyMISP method*), 40, 85
 event_blocklists() (*pymisp.PyMISP method*), 40, 85
 event_delegations() (*pymisp.PyMISP method*), 40, 85
 event_exists() (*pymisp.PyMISP method*), 40, 86
 events() (*pymisp.PyMISP method*), 40, 86
 ExpandedPyMISP (*class in pymisp*), 10
- ## F
- Fail2BanObject (*class in pymisp.tools*), 154
 feeds() (*pymisp.PyMISP method*), 41, 86
 fetch_feed() (*pymisp.PyMISP method*), 41, 86
 FileObject (*class in pymisp.tools*), 154, 158
 fork_galaxy_cluster() (*pymisp.PyMISP method*), 41, 86
 freetext() (*pymisp.PyMISP method*), 41, 87
 from_dict() (*pymisp.AbstractMISP method*), 9, 113
 from_dict() (*pymisp.MISPAttribute method*), 10, 122
 from_dict() (*pymisp.MISPCorrelationExclusion method*), 11
 from_dict() (*pymisp.MISPDecayingModel method*), 11
 from_dict() (*pymisp.MISPEvent method*), 12, 116
 from_dict() (*pymisp.MISPEventBlocklist method*), 15, 119
 from_dict() (*pymisp.MISPEventDelegation method*), 15, 120
 from_dict() (*pymisp.MISPEventReport method*), 15
 from_dict() (*pymisp.MISPFeed method*), 15, 137
 from_dict() (*pymisp.MISPGalaxy method*), 15
 from_dict() (*pymisp.MISPGalaxyCluster method*), 16
 from_dict() (*pymisp.MISPGalaxyClusterElement method*), 17
 from_dict() (*pymisp.MISPGalaxyClusterRelation method*), 17
 from_dict() (*pymisp.MISPInbox method*), 17, 138
 from_dict() (*pymisp.MISPLog method*), 18, 140
 from_dict() (*pymisp.MISPNote method*), 18
 from_dict() (*pymisp.MISPNoticelist method*), 18, 141
 from_dict() (*pymisp.MISPObject method*), 19, 124
 from_dict() (*pymisp.MISPObjectAttribute method*), 20, 127
 from_dict() (*pymisp.MISPObjectReference method*), 20, 128
 from_dict() (*pymisp.MISPObjectTemplate method*), 20, 130
 from_dict() (*pymisp.MISPOpinion method*), 20
 from_dict() (*pymisp.MISPOrganisation method*), 20, 135
 from_dict() (*pymisp.MISPOrganisationBlocklist method*), 21, 136
 from_dict() (*pymisp.MISPRelationship method*), 21
 from_dict() (*pymisp.MISPRole method*), 21, 142
 from_dict() (*pymisp.MISPServer method*), 21, 144
 from_dict() (*pymisp.MISPShadowAttribute method*), 21, 145
 from_dict() (*pymisp.MISPSharingGroup method*), 21, 146
 from_dict() (*pymisp.MISPSighting method*), 22, 147
 from_dict() (*pymisp.MISPTag method*), 22, 131
 from_dict() (*pymisp.MISPTaxonomy method*), 22, 149
 from_dict() (*pymisp.MISPUser method*), 22, 132
 from_dict() (*pymisp.MISPUserSetting method*), 22, 133
 from_dict() (*pymisp.MISPWarninglist method*), 23, 150
 from_dict() (*pymisp.tools.ELFObject method*), 161
 from_dict() (*pymisp.tools.ELFSectionObject method*),

163
 from_dict() (*pymisp.tools.FileObject* method), 158
 from_dict() (*pymisp.tools.MachOObject* method), 170
 from_dict() (*pymisp.tools.MachOSectionObject* method), 173
 from_dict() (*pymisp.tools.PEObject* method), 166
 from_dict() (*pymisp.tools.PESectionObject* method), 168
 from_dict() (*pymisp.tools.VTReportObject* method), 175
 from_json() (*pymisp.AbstractMISP* method), 9, 114
 from_json() (*pymisp.MISPAttribute* method), 122
 from_json() (*pymisp.MISPEvent* method), 116
 from_json() (*pymisp.MISPEventBlocklist* method), 119
 from_json() (*pymisp.MISPEventDelegation* method), 121
 from_json() (*pymisp.MISPFeed* method), 137
 from_json() (*pymisp.MISPInbox* method), 139
 from_json() (*pymisp.MISPLog* method), 140
 from_json() (*pymisp.MISPNoticelist* method), 141
 from_json() (*pymisp.MISPObject* method), 125
 from_json() (*pymisp.MISPObjectAttribute* method), 127
 from_json() (*pymisp.MISPObjectReference* method), 128
 from_json() (*pymisp.MISPObjectTemplate* method), 130
 from_json() (*pymisp.MISPOrganisation* method), 135
 from_json() (*pymisp.MISPOrganisationBlocklist* method), 136
 from_json() (*pymisp.MISPRole* method), 142
 from_json() (*pymisp.MISPServer* method), 144
 from_json() (*pymisp.MISPShadowAttribute* method), 145
 from_json() (*pymisp.MISPSharingGroup* method), 146
 from_json() (*pymisp.MISPSighting* method), 148
 from_json() (*pymisp.MISPTag* method), 131
 from_json() (*pymisp.MISPTaxonomy* method), 149
 from_json() (*pymisp.MISPUser* method), 132
 from_json() (*pymisp.MISPUserSetting* method), 133
 from_json() (*pymisp.MISPWarninglist* method), 150
 from_json() (*pymisp.tools.ELFObject* method), 161
 from_json() (*pymisp.tools.ELFSectionObject* method), 163
 from_json() (*pymisp.tools.FileObject* method), 159
 from_json() (*pymisp.tools.MachOObject* method), 170
 from_json() (*pymisp.tools.MachOSectionObject* method), 173
 from_json() (*pymisp.tools.PEObject* method), 166
 from_json() (*pymisp.tools.PESectionObject* method), 168
 from_json() (*pymisp.tools.VTReportObject* method), 175

G

galaxies (*pymisp.MISPAttribute* property), 10, 122
 galaxies (*pymisp.MISPObjectAttribute* property), 127
 galaxies() (*pymisp.PyMISP* method), 42, 87
 generate_attributes() (*pymisp.tools.AbstractMISPObjectGenerator* method), 153
 generate_attributes() (*pymisp.tools.ASNObject* method), 153
 generate_attributes() (*pymisp.tools.DataURLObject* method), 153
 generate_attributes() (*pymisp.tools.DomainIPOObject* method), 153
 generate_attributes() (*pymisp.tools.ELFObject* method), 153, 161
 generate_attributes() (*pymisp.tools.ELFSectionObject* method), 153, 163
 generate_attributes() (*pymisp.tools.Fail2BanObject* method), 154
 generate_attributes() (*pymisp.tools.FileObject* method), 154, 159
 generate_attributes() (*pymisp.tools.GenericObjectGenerator* method), 154
 generate_attributes() (*pymisp.tools.GeolocationObject* method), 154
 generate_attributes() (*pymisp.tools.GitVulnFinderObject* method), 154
 generate_attributes() (*pymisp.tools.MachOObject* method), 155, 171
 generate_attributes() (*pymisp.tools.MachOSectionObject* method), 155, 173
 generate_attributes() (*pymisp.tools.PEObject* method), 155, 166
 generate_attributes() (*pymisp.tools.PESectionObject* method), 155, 168
 generate_attributes() (*pymisp.tools.SBSignatureObject* method), 155
 generate_attributes() (*pymisp.tools.SSHAuthorizedKeysObject* method), 155
 generate_attributes() (*pymisp.tools.VehicleObject* method), 156
 generate_attributes() (*pymisp.tools.VTReportObject* method), 155, 175

- GenericObjectGenerator (class in pymisp.tools), 154
- GeolocationObject (class in pymisp.tools), 154
- get() (pymisp.MISPAttribute method), 123
- get() (pymisp.MISPEvent method), 116
- get() (pymisp.MISPEventBlocklist method), 119
- get() (pymisp.MISPEventDelegation method), 121
- get() (pymisp.MISPFeed method), 137
- get() (pymisp.MISPInbox method), 139
- get() (pymisp.MISPLog method), 140
- get() (pymisp.MISPNoticelist method), 141
- get() (pymisp.MISPObject method), 125
- get() (pymisp.MISPObjectAttribute method), 127
- get() (pymisp.MISPObjectReference method), 128
- get() (pymisp.MISPObjectTemplate method), 130
- get() (pymisp.MISPOrganisation method), 135
- get() (pymisp.MISPOrganisationBlocklist method), 136
- get() (pymisp.MISPRole method), 143
- get() (pymisp.MISPServer method), 144
- get() (pymisp.MISPShadowAttribute method), 145
- get() (pymisp.MISPSharingGroup method), 146
- get() (pymisp.MISP sighting method), 148
- get() (pymisp.MISPTag method), 131
- get() (pymisp.MISPTaxonomy method), 149
- get() (pymisp.MISPUser method), 132
- get() (pymisp.MISPUserSetting method), 134
- get() (pymisp.MISPWarninglist method), 150
- get() (pymisp.tools.ELFObject method), 161
- get() (pymisp.tools.ELFSectionObject method), 163
- get() (pymisp.tools.FileObject method), 159
- get() (pymisp.tools.MachOObject method), 171
- get() (pymisp.tools.MachOSectionObject method), 173
- get() (pymisp.tools.PEObject method), 166
- get() (pymisp.tools.PESectionObject method), 168
- get() (pymisp.tools.VTReportObject method), 176
- get_all_functions() (pymisp.PyMISP method), 42, 87
- get_analyst_data() (pymisp.PyMISP method), 42, 87
- get_attribute() (pymisp.PyMISP method), 42, 87
- get_attribute_by_id() (pymisp.MISPEvent method), 13, 116
- get_attribute_by_id() (pymisp.MISPObject method), 19, 125
- get_attribute_by_id() (pymisp.tools.ELFObject method), 161
- get_attribute_by_id() (pymisp.tools.ELFSectionObject method), 164
- get_attribute_by_id() (pymisp.tools.FileObject method), 159
- get_attribute_by_id() (pymisp.tools.MachOObject method), 171
- get_attribute_by_id() (pymisp.tools.MachOSectionObject method), 173
- get_attribute_by_id() (pymisp.tools.PEObject method), 166
- get_attribute_by_id() (pymisp.tools.PESectionObject method), 168
- get_attribute_by_id() (pymisp.tools.VTReportObject method), 176
- get_attribute_by_uuid() (pymisp.MISPEvent method), 13, 116
- get_attribute_by_uuid() (pymisp.MISPObject method), 19, 125
- get_attribute_by_uuid() (pymisp.tools.ELFObject method), 161
- get_attribute_by_uuid() (pymisp.tools.ELFSectionObject method), 164
- get_attribute_by_uuid() (pymisp.tools.FileObject method), 159
- get_attribute_by_uuid() (pymisp.tools.MachOObject method), 171
- get_attribute_by_uuid() (pymisp.tools.MachOSectionObject method), 173
- get_attribute_by_uuid() (pymisp.tools.PEObject method), 166
- get_attribute_by_uuid() (pymisp.tools.PESectionObject method), 168
- get_attribute_by_uuid() (pymisp.tools.VTReportObject method), 176
- get_attribute_proposal() (pymisp.PyMISP method), 42, 87
- get_attribute_tag() (pymisp.MISPEvent method), 13, 117
- get_attributes_by_relation() (pymisp.MISPObject method), 19, 125
- get_attributes_by_relation() (pymisp.tools.ELFObject method), 161
- get_attributes_by_relation() (pymisp.tools.ELFSectionObject method), 164
- get_attributes_by_relation() (pymisp.tools.FileObject method), 159
- get_attributes_by_relation() (pymisp.tools.MachOObject method), 171
- get_attributes_by_relation() (pymisp.tools.MachOSectionObject method), 173
- get_attributes_by_relation() (pymisp.tools.PEObject method), 166
- get_attributes_by_relation() (pymisp.tools.PESectionObject method), 168

- 169
 get_attributes_by_relation()
 (*pymisp.tools.VTReportObject* method), 176
 get_community() (*pymisp.PyMISP* method), 42, 88
 get_correlation_exclusion() (*pymisp.PyMISP* method), 43, 88
 get_event() (*pymisp.PyMISP* method), 43, 88
 get_event_report() (*pymisp.PyMISP* method), 43, 88
 get_event_reports() (*pymisp.PyMISP* method), 43, 89
 get_feed() (*pymisp.PyMISP* method), 43, 89
 get_galaxy() (*pymisp.PyMISP* method), 44, 89
 get_galaxy_cluster() (*pymisp.PyMISP* method), 44, 89
 get_new_authkey() (*pymisp.PyMISP* method), 44, 89
 get_note() (*pymisp.PyMISP* method), 44, 89
 get_noticelist() (*pymisp.PyMISP* method), 44, 90
 get_object() (*pymisp.PyMISP* method), 45, 90
 get_object_by_id() (*pymisp.MISPEvent* method), 13, 117
 get_object_by_uuid() (*pymisp.MISPEvent* method), 13, 117
 get_object_template() (*pymisp.PyMISP* method), 45, 90
 get_objects_by_name() (*pymisp.MISPEvent* method), 13, 117
 get_opinion() (*pymisp.PyMISP* method), 45, 90
 get_organisation() (*pymisp.PyMISP* method), 45, 90
 get_raw_object_template() (*pymisp.PyMISP* method), 45, 91
 get_relationship() (*pymisp.PyMISP* method), 45, 91
 get_server_setting() (*pymisp.PyMISP* method), 46, 91
 get_sharing_group() (*pymisp.PyMISP* method), 46, 91
 get_sync_config() (*pymisp.PyMISP* method), 46, 91
 get_tag() (*pymisp.PyMISP* method), 46, 91
 get_taxonomy() (*pymisp.PyMISP* method), 46, 92
 get_user() (*pymisp.PyMISP* method), 46, 92
 get_user_setting() (*pymisp.PyMISP* method), 47, 92
 get_warninglist() (*pymisp.PyMISP* method), 47, 92
 get_workers() (*pymisp.PyMISP* method), 47, 93
 GitVulnFinderObject (*class in pymisp.tools*), 154
- ## H
- has_attributes_by_relation()
 (*pymisp.MISPObject* method), 19, 125
 has_attributes_by_relation()
 (*pymisp.tools.ELFObject* method), 162
 has_attributes_by_relation()
 (*pymisp.tools.ELFSectionObject* method), 164
 has_attributes_by_relation()
 (*pymisp.tools.FileObject* method), 159
 has_attributes_by_relation()
 (*pymisp.tools.MachObject* method), 171
 has_attributes_by_relation()
 (*pymisp.tools.MachOSectionObject* method), 173
 has_attributes_by_relation()
 (*pymisp.tools.PEObject* method), 166
 has_attributes_by_relation()
 (*pymisp.tools.PESectionObject* method), 169
 has_attributes_by_relation()
 (*pymisp.tools.VTReportObject* method), 176
 hash_values() (*pymisp.MISPAttribute* method), 10, 123
 hash_values() (*pymisp.MISPObjectAttribute* method), 127
- ## I
- import_server() (*pymisp.PyMISP* method), 47, 93
 InvalidMISPObject, 10
 items() (*pymisp.MISPAttribute* method), 123
 items() (*pymisp.MISPEvent* method), 117
 items() (*pymisp.MISPEventBlocklist* method), 119
 items() (*pymisp.MISPEventDelegation* method), 121
 items() (*pymisp.MISPFeed* method), 137
 items() (*pymisp.MISPInbox* method), 139
 items() (*pymisp.MISPLog* method), 140
 items() (*pymisp.MISPNoticelist* method), 141
 items() (*pymisp.MISPObject* method), 125
 items() (*pymisp.MISPObjectAttribute* method), 127
 items() (*pymisp.MISPObjectReference* method), 129
 items() (*pymisp.MISPObjectTemplate* method), 130
 items() (*pymisp.MISPOrganisation* method), 135
 items() (*pymisp.MISPOrganisationBlocklist* method), 136
 items() (*pymisp.MISPRole* method), 143
 items() (*pymisp.MISPServer* method), 144
 items() (*pymisp.MISPShadowAttribute* method), 145
 items() (*pymisp.MISPSharingGroup* method), 146
 items() (*pymisp.MISPSighting* method), 148
 items() (*pymisp.MISPTag* method), 131
 items() (*pymisp.MISPTaxonomy* method), 149
 items() (*pymisp.MISPUser* method), 132
 items() (*pymisp.MISPUserSetting* method), 134
 items() (*pymisp.MISPWarninglist* method), 150
 items() (*pymisp.tools.ELFObject* method), 162
 items() (*pymisp.tools.ELFSectionObject* method), 164
 items() (*pymisp.tools.FileObject* method), 159
 items() (*pymisp.tools.MachObject* method), 171
 items() (*pymisp.tools.MachOSectionObject* method), 174
 items() (*pymisp.tools.PEObject* method), 166

items() (*pymisp.tools.PESectionObject* method), 169
 items() (*pymisp.tools.VTReportObject* method), 176

J

jsonable() (*pymisp.AbstractMISP* method), 9, 114
 jsonable() (*pymisp.MISPAttribute* method), 123
 jsonable() (*pymisp.MISPEvent* method), 117
 jsonable() (*pymisp.MISPEventBlocklist* method), 119
 jsonable() (*pymisp.MISPEventDelegation* method), 121
 jsonable() (*pymisp.MISPFeed* method), 138
 jsonable() (*pymisp.MISPInbox* method), 139
 jsonable() (*pymisp.MISPLog* method), 140
 jsonable() (*pymisp.MISPNoticelist* method), 141
 jsonable() (*pymisp.MISPObject* method), 125
 jsonable() (*pymisp.MISPObjectAttribute* method), 127
 jsonable() (*pymisp.MISPObjectReference* method), 129
 jsonable() (*pymisp.MISPObjectTemplate* method), 130
 jsonable() (*pymisp.MISPOrganisation* method), 135
 jsonable() (*pymisp.MISPOrganisationBlocklist* method), 136
 jsonable() (*pymisp.MISPRole* method), 143
 jsonable() (*pymisp.MISPServer* method), 144
 jsonable() (*pymisp.MISPShadowAttribute* method), 145
 jsonable() (*pymisp.MISPSharingGroup* method), 147
 jsonable() (*pymisp.MISPSighting* method), 148
 jsonable() (*pymisp.MISPTag* method), 131
 jsonable() (*pymisp.MISPTaxonomy* method), 149
 jsonable() (*pymisp.MISPUser* method), 132
 jsonable() (*pymisp.MISPUserSetting* method), 134
 jsonable() (*pymisp.MISPWarninglist* method), 151
 jsonable() (*pymisp.tools.ELFObject* method), 162
 jsonable() (*pymisp.tools.ELFSectionObject* method), 164
 jsonable() (*pymisp.tools.FileObject* method), 159
 jsonable() (*pymisp.tools.MachOObject* method), 171
 jsonable() (*pymisp.tools.MachOSectionObject* method), 174
 jsonable() (*pymisp.tools.PEObject* method), 166
 jsonable() (*pymisp.tools.PESectionObject* method), 169
 jsonable() (*pymisp.tools.VTReportObject* method), 176

K

keys() (*pymisp.MISPAttribute* method), 123
 keys() (*pymisp.MISPEvent* method), 117
 keys() (*pymisp.MISPEventBlocklist* method), 120
 keys() (*pymisp.MISPEventDelegation* method), 121
 keys() (*pymisp.MISPFeed* method), 138
 keys() (*pymisp.MISPInbox* method), 139
 keys() (*pymisp.MISPLog* method), 140

keys() (*pymisp.MISPNoticelist* method), 141
 keys() (*pymisp.MISPObject* method), 125
 keys() (*pymisp.MISPObjectAttribute* method), 127
 keys() (*pymisp.MISPObjectReference* method), 129
 keys() (*pymisp.MISPObjectTemplate* method), 130
 keys() (*pymisp.MISPOrganisation* method), 135
 keys() (*pymisp.MISPOrganisationBlocklist* method), 136
 keys() (*pymisp.MISPRole* method), 143
 keys() (*pymisp.MISPServer* method), 144
 keys() (*pymisp.MISPShadowAttribute* method), 145
 keys() (*pymisp.MISPSharingGroup* method), 147
 keys() (*pymisp.MISPSighting* method), 148
 keys() (*pymisp.MISPTag* method), 131
 keys() (*pymisp.MISPTaxonomy* method), 149
 keys() (*pymisp.MISPUser* method), 132
 keys() (*pymisp.MISPUserSetting* method), 134
 keys() (*pymisp.MISPWarninglist* method), 151
 keys() (*pymisp.tools.ELFObject* method), 162
 keys() (*pymisp.tools.ELFSectionObject* method), 164
 keys() (*pymisp.tools.FileObject* method), 159
 keys() (*pymisp.tools.MachOObject* method), 171
 keys() (*pymisp.tools.MachOSectionObject* method), 174
 keys() (*pymisp.tools.PEObject* method), 167
 keys() (*pymisp.tools.PESectionObject* method), 169
 keys() (*pymisp.tools.VTReportObject* method), 176
 kill_all_workers() (*pymisp.PyMISP* method), 47, 93
 known_types (*pymisp.MISPAttribute* property), 11, 123
 known_types (*pymisp.MISPObjectAttribute* property), 127

L

load() (*pymisp.MISPEvent* method), 13, 117
 load_default_feeds() (*pymisp.PyMISP* method), 48, 93
 load_file() (*pymisp.MISPEvent* method), 14, 117
 load_openioc() (*pymisp.tools* method), 177
 load_openioc_file() (*pymisp.tools* method), 177

M

Mach0Object (*class in pymisp.tools*), 154, 170
 Mach0SectionObject (*class in pymisp.tools*), 155, 172
 malware_binary (*pymisp.MISPAttribute* property), 11, 123
 malware_binary (*pymisp.MISPObjectAttribute* property), 127
 misp_instance_version (*pymisp.PyMISP* property), 48, 93
 misp_instance_version_master (*pymisp.PyMISP* property), 48, 93
 MISPAttribute (*class in pymisp*), 10, 122
 MISPCorrelationExclusion (*class in pymisp*), 11
 MISPDecayingModel (*class in pymisp*), 11
 MISPEncode (*class in pymisp*), 114

MISPEvent (class in pymisp), 11, 115
 MISPEventBlocklist (class in pymisp), 15, 119
 MISPEventDelegation (class in pymisp), 15, 120
 MISPEventReport (class in pymisp), 15
 MISPFeeD (class in pymisp), 15, 137
 MISPGalaxy (class in pymisp), 15
 MISPGalaxyCluster (class in pymisp), 16
 MISPGalaxyClusterElement (class in pymisp), 17
 MISPGalaxyClusterRelation (class in pymisp), 17
 MISPInbox (class in pymisp), 17, 138
 MISPLog (class in pymisp), 18, 140
 MISPNote (class in pymisp), 18
 MISPNoticelist (class in pymisp), 18, 141
 MISPObject (class in pymisp), 18, 124
 MISPObjectAttribute (class in pymisp), 20, 126
 MISPObjectException, 20
 MISPObjectReference (class in pymisp), 20, 128
 MISPObjectTemplate (class in pymisp), 20, 129
 MISPOpinion (class in pymisp), 20
 MISPOrganisation (class in pymisp), 20, 135
 MISPOrganisationBlocklist (class in pymisp), 20, 136
 MISPRelationship (class in pymisp), 21
 MISPRole (class in pymisp), 21, 142
 MISPServer (class in pymisp), 21, 144
 MISPServerError, 21
 MISPShadowAttribute (class in pymisp), 21, 145
 MISPSharingGroup (class in pymisp), 21, 146
 MISPSighting (class in pymisp), 22, 147
 MISPTag (class in pymisp), 22, 131
 MISPTaxonomy (class in pymisp), 22, 149
 MISPUser (class in pymisp), 22, 132
 MISPUserSetting (class in pymisp), 22, 133
 MISPWarninglist (class in pymisp), 23, 150
 module
 pymisp, 9
 pymisp.tools, 153

N

NewAttributeError, 23
 NewEventError, 23
 NoKey, 23
 noticelists() (pymisp.PyMISP method), 48, 93
 NoURL, 23

O

object_exists() (pymisp.PyMISP method), 48, 93
 object_templates() (pymisp.PyMISP method), 48, 93
 organisation_blocklists() (pymisp.PyMISP method), 48, 94
 organisation_exists() (pymisp.PyMISP method), 48, 94
 organisations() (pymisp.PyMISP method), 49, 94

P

parse_meta_as_elements()
 (pymisp.MISPGalaxyCluster method), 17
 PEObject (class in pymisp.tools), 155, 165
 PESectionObject (class in pymisp.tools), 155, 167
 pop() (pymisp.MISPAttribute method), 123
 pop() (pymisp.MISPEvent method), 118
 pop() (pymisp.MISPEventBlocklist method), 120
 pop() (pymisp.MISPEventDelegation method), 121
 pop() (pymisp.MISPFeeD method), 138
 pop() (pymisp.MISPInbox method), 139
 pop() (pymisp.MISPLog method), 140
 pop() (pymisp.MISPNoticelist method), 142
 pop() (pymisp.MISPObject method), 125
 pop() (pymisp.MISPObjectAttribute method), 127
 pop() (pymisp.MISPObjectReference method), 129
 pop() (pymisp.MISPObjectTemplate method), 130
 pop() (pymisp.MISPOrganisation method), 135
 pop() (pymisp.MISPOrganisationBlocklist method), 136
 pop() (pymisp.MISPRole method), 143
 pop() (pymisp.MISPServer method), 144
 pop() (pymisp.MISPShadowAttribute method), 145
 pop() (pymisp.MISPSharingGroup method), 147
 pop() (pymisp.MISPSighting method), 148
 pop() (pymisp.MISPTag method), 131
 pop() (pymisp.MISPTaxonomy method), 149
 pop() (pymisp.MISPUser method), 133
 pop() (pymisp.MISPUserSetting method), 134
 pop() (pymisp.MISPWarninglist method), 151
 pop() (pymisp.tools.ELFObject method), 162
 pop() (pymisp.tools.ELFSectionObject method), 164
 pop() (pymisp.tools.FileObject method), 159
 pop() (pymisp.tools.MachOObject method), 171
 pop() (pymisp.tools.MachOSectionObject method), 174
 pop() (pymisp.tools.PEObject method), 167
 pop() (pymisp.tools.PESectionObject method), 169
 pop() (pymisp.tools.VTReportObject method), 176
 popitem() (pymisp.MISPAttribute method), 123
 popitem() (pymisp.MISPEvent method), 118
 popitem() (pymisp.MISPEventBlocklist method), 120
 popitem() (pymisp.MISPEventDelegation method), 121
 popitem() (pymisp.MISPFeeD method), 138
 popitem() (pymisp.MISPInbox method), 139
 popitem() (pymisp.MISPLog method), 140
 popitem() (pymisp.MISPNoticelist method), 142
 popitem() (pymisp.MISPObject method), 125
 popitem() (pymisp.MISPObjectAttribute method), 127
 popitem() (pymisp.MISPObjectReference method), 129
 popitem() (pymisp.MISPObjectTemplate method), 130
 popitem() (pymisp.MISPOrganisation method), 135
 popitem() (pymisp.MISPOrganisationBlocklist method), 136
 popitem() (pymisp.MISPRole method), 143
 popitem() (pymisp.MISPServer method), 144

popitem() (*pymisp.MISPShadowAttribute method*), 145
 popitem() (*pymisp.MISPSharingGroup method*), 147
 popitem() (*pymisp.MISPSighting method*), 148
 popitem() (*pymisp.MISPTag method*), 131
 popitem() (*pymisp.MISPTaxonomy method*), 149
 popitem() (*pymisp.MISPUser method*), 133
 popitem() (*pymisp.MISPUserSetting method*), 134
 popitem() (*pymisp.MISPWarninglist method*), 151
 popitem() (*pymisp.tools.ELFObject method*), 162
 popitem() (*pymisp.tools.ELFSectionObject method*), 164
 popitem() (*pymisp.tools.FileObject method*), 159
 popitem() (*pymisp.tools.MachObject method*), 171
 popitem() (*pymisp.tools.MachOSectionObject method*), 174
 popitem() (*pymisp.tools.PEObject method*), 167
 popitem() (*pymisp.tools.PESectionObject method*), 169
 popitem() (*pymisp.tools.VTReportObject method*), 176
 publish() (*pymisp.MISPEvent method*), 14, 118
 publish() (*pymisp.PyMISP method*), 49, 94
 publish_galaxy_cluster() (*pymisp.PyMISP method*), 49, 94
 push_event_to_ZMQ() (*pymisp.PyMISP method*), 49, 95
 pymisp
 module, 9
 PyMISP (*class in pymisp*), 23, 68
 pymisp.tools
 module, 153
 pymisp_version_main (*pymisp.PyMISP property*), 49, 95
 pymisp_version_master (*pymisp.PyMISP property*), 49, 95
 PyMISPError, 68
 PyMISPInvalidFormat, 68

R

recommended_pymisp_version (*pymisp.PyMISP property*), 49, 95
 register_user() (*in module pymisp*), 68
 remote_acl() (*pymisp.PyMISP method*), 49, 95
 remove_org_from_sharing_group() (*pymisp.PyMISP method*), 50, 95
 remove_server_from_sharing_group() (*pymisp.PyMISP method*), 50, 95
 request_community_access() (*pymisp.PyMISP method*), 50, 95
 restart_dead_workers() (*pymisp.PyMISP method*), 51, 96
 restart_workers() (*pymisp.PyMISP method*), 51, 96
 restore_attribute() (*pymisp.PyMISP method*), 51, 96
 roles() (*pymisp.PyMISP method*), 51, 96

S

SBSignatureObject (*class in pymisp.tools*), 155
 search() (*pymisp.PyMISP method*), 51, 96
 search_feeds() (*pymisp.PyMISP method*), 54, 99
 search_galaxy() (*pymisp.PyMISP method*), 54, 99
 search_galaxy_clusters() (*pymisp.PyMISP method*), 54, 99
 search_index() (*pymisp.PyMISP method*), 54, 100
 search_logs() (*pymisp.PyMISP method*), 56, 101
 search_sightings() (*pymisp.PyMISP method*), 56, 102
 search_tags() (*pymisp.PyMISP method*), 57, 103
 server_pull() (*pymisp.PyMISP method*), 57, 103
 server_push() (*pymisp.PyMISP method*), 58, 103
 server_settings() (*pymisp.PyMISP method*), 58, 103
 servers() (*pymisp.PyMISP method*), 58, 103
 set_date() (*pymisp.MISPEvent method*), 14, 118
 set_default_role() (*pymisp.PyMISP method*), 58, 104
 set_not_jsonable() (*pymisp.AbstractMISP method*), 9, 114
 set_not_jsonable() (*pymisp.MISPAttribute method*), 123
 set_not_jsonable() (*pymisp.MISPEvent method*), 118
 set_not_jsonable() (*pymisp.MISPEventBlocklist method*), 120
 set_not_jsonable() (*pymisp.MISPEventDelegation method*), 121
 set_not_jsonable() (*pymisp.MISPFeed method*), 138
 set_not_jsonable() (*pymisp.MISPInbox method*), 139
 set_not_jsonable() (*pymisp.MISPLog method*), 140
 set_not_jsonable() (*pymisp.MISPNoticelist method*), 142
 set_not_jsonable() (*pymisp.MISPObject method*), 125
 set_not_jsonable() (*pymisp.MISPObjectAttribute method*), 127
 set_not_jsonable() (*pymisp.MISPObjectReference method*), 129
 set_not_jsonable() (*pymisp.MISPObjectTemplate method*), 130
 set_not_jsonable() (*pymisp.MISPOrganisation method*), 135
 set_not_jsonable() (*pymisp.MISPOrganisationBlocklist method*), 136
 set_not_jsonable() (*pymisp.MISPRole method*), 143
 set_not_jsonable() (*pymisp.MISPServer method*), 144
 set_not_jsonable() (*pymisp.MISPShadowAttribute method*), 145
 set_not_jsonable() (*pymisp.MISPSharingGroup method*), 147
 set_not_jsonable() (*pymisp.MISPSighting method*), 148

- set_not_jsonable() (*pymisp.MISPTag method*), 131
 set_not_jsonable() (*pymisp.MISPTaxonomy method*), 149
 set_not_jsonable() (*pymisp.MISPUser method*), 133
 set_not_jsonable() (*pymisp.MISPUserSetting method*), 134
 set_not_jsonable() (*pymisp.MISPWarninglist method*), 151
 set_not_jsonable() (*pymisp.tools.ELFObject method*), 162
 set_not_jsonable() (*pymisp.tools.ELFSectionObject method*), 164
 set_not_jsonable() (*pymisp.tools.FileObject method*), 160
 set_not_jsonable() (*pymisp.tools.MachOObject method*), 171
 set_not_jsonable() (*pymisp.tools.MachOSectionObject method*), 174
 set_not_jsonable() (*pymisp.tools.PEObject method*), 167
 set_not_jsonable() (*pymisp.tools.PESectionObject method*), 169
 set_not_jsonable() (*pymisp.tools.VTReportObject method*), 176
 set_server_setting() (*pymisp.PyMISP method*), 58, 104
 set_user_setting() (*pymisp.PyMISP method*), 58, 104
 setdefault() (*pymisp.MISPAttribute method*), 123
 setdefault() (*pymisp.MISPEvent method*), 118
 setdefault() (*pymisp.MISPEventBlocklist method*), 120
 setdefault() (*pymisp.MISPEventDelegation method*), 121
 setdefault() (*pymisp.MISPFeed method*), 138
 setdefault() (*pymisp.MISPInbox method*), 139
 setdefault() (*pymisp.MISPLog method*), 140
 setdefault() (*pymisp.MISPNoticelist method*), 142
 setdefault() (*pymisp.MISPObject method*), 126
 setdefault() (*pymisp.MISPObjectAttribute method*), 128
 setdefault() (*pymisp.MISPObjectReference method*), 129
 setdefault() (*pymisp.MISPObjectTemplate method*), 130
 setdefault() (*pymisp.MISPOrganisation method*), 135
 setdefault() (*pymisp.MISPOrganisationBlocklist method*), 137
 setdefault() (*pymisp.MISPRole method*), 143
 setdefault() (*pymisp.MISPServer method*), 144
 setdefault() (*pymisp.MISPShadowAttribute method*), 146
 setdefault() (*pymisp.MISPSharingGroup method*), 147
 setdefault() (*pymisp.MISPSighting method*), 148
 setdefault() (*pymisp.MISPTag method*), 131
 setdefault() (*pymisp.MISPTaxonomy method*), 150
 setdefault() (*pymisp.MISPUser method*), 133
 setdefault() (*pymisp.MISPUserSetting method*), 134
 setdefault() (*pymisp.MISPWarninglist method*), 151
 setdefault() (*pymisp.tools.ELFObject method*), 162
 setdefault() (*pymisp.tools.ELFSectionObject method*), 164
 setdefault() (*pymisp.tools.FileObject method*), 160
 setdefault() (*pymisp.tools.MachOObject method*), 172
 setdefault() (*pymisp.tools.MachOSectionObject method*), 174
 setdefault() (*pymisp.tools.PEObject method*), 167
 setdefault() (*pymisp.tools.PESectionObject method*), 169
 setdefault() (*pymisp.tools.VTReportObject method*), 176
 sharing_group_exists() (*pymisp.PyMISP method*), 59, 104
 sharing_groups() (*pymisp.PyMISP method*), 59, 104
 sightings() (*pymisp.PyMISP method*), 59, 104
 sign_blob() (*pymisp.PyMISP method*), 59, 105
 SSHAuthorizedKeysObject (*class in pymisp.tools*), 155
 start_worker() (*pymisp.PyMISP method*), 59, 105
 stop_worker_by_pid() (*pymisp.PyMISP method*), 59, 105
- ## T
- tag() (*pymisp.PyMISP method*), 60, 105
 tags (*pymisp.MISPAttribute property*), 11, 123
 tags (*pymisp.MISPEvent property*), 14, 118
 tags (*pymisp.MISPGalaxyClusterRelation property*), 17
 tags (*pymisp.MISPObjectAttribute property*), 128
 tags() (*pymisp.PyMISP method*), 60, 105
 tags_statistics() (*pymisp.PyMISP method*), 60, 106
 taxonomies() (*pymisp.PyMISP method*), 60, 106
 test_server() (*pymisp.PyMISP method*), 60, 106
 ThreatLevel (*class in pymisp*), 68
 to_dict() (*pymisp.AbstractMISP method*), 9, 114
 to_dict() (*pymisp.MISPAttribute method*), 11, 123
 to_dict() (*pymisp.MISPEvent method*), 14, 118
 to_dict() (*pymisp.MISPEventBlocklist method*), 120
 to_dict() (*pymisp.MISPEventDelegation method*), 121
 to_dict() (*pymisp.MISPFeed method*), 138
 to_dict() (*pymisp.MISPInbox method*), 139
 to_dict() (*pymisp.MISPLog method*), 140
 to_dict() (*pymisp.MISPNoticelist method*), 142
 to_dict() (*pymisp.MISPObject method*), 19, 126
 to_dict() (*pymisp.MISPObjectAttribute method*), 128
 to_dict() (*pymisp.MISPObjectReference method*), 129
 to_dict() (*pymisp.MISPObjectTemplate method*), 130
 to_dict() (*pymisp.MISPOrganisation method*), 135

- to_dict() (*pymisp.MISPOrganisationBlocklist method*), 137
- to_dict() (*pymisp.MISPRole method*), 143
- to_dict() (*pymisp.MISPServer method*), 144
- to_dict() (*pymisp.MISPShadowAttribute method*), 146
- to_dict() (*pymisp.MISPSharingGroup method*), 147
- to_dict() (*pymisp.MISPSighting method*), 148
- to_dict() (*pymisp.MISPTag method*), 131
- to_dict() (*pymisp.MISPTaxonomy method*), 150
- to_dict() (*pymisp.MISPUser method*), 133
- to_dict() (*pymisp.MISPUserSetting method*), 134
- to_dict() (*pymisp.MISPWarninglist method*), 151
- to_dict() (*pymisp.tools.ELFObject method*), 162
- to_dict() (*pymisp.tools.ELFSectionObject method*), 164
- to_dict() (*pymisp.tools.FileObject method*), 160
- to_dict() (*pymisp.tools.MachOObject method*), 172
- to_dict() (*pymisp.tools.MachOSectionObject method*), 174
- to_dict() (*pymisp.tools.PEObject method*), 167
- to_dict() (*pymisp.tools.PESectionObject method*), 169
- to_dict() (*pymisp.tools.VTReportObject method*), 176
- to_feed() (*pymisp.MISPEvent method*), 14, 118
- to_json() (*pymisp.AbstractMISP method*), 9, 114
- to_json() (*pymisp.MISPAAttribute method*), 123
- to_json() (*pymisp.MISPEvent method*), 119
- to_json() (*pymisp.MISPEventBlocklist method*), 120
- to_json() (*pymisp.MISPEventDelegation method*), 121
- to_json() (*pymisp.MISPFeed method*), 138
- to_json() (*pymisp.MISPInbox method*), 139
- to_json() (*pymisp.MISPLog method*), 141
- to_json() (*pymisp.MISPNoticelist method*), 142
- to_json() (*pymisp.MISPObject method*), 19, 126
- to_json() (*pymisp.MISPObjectAttribute method*), 128
- to_json() (*pymisp.MISPObjectReference method*), 129
- to_json() (*pymisp.MISPObjectTemplate method*), 130
- to_json() (*pymisp.MISPOrganisation method*), 135
- to_json() (*pymisp.MISPOrganisationBlocklist method*), 137
- to_json() (*pymisp.MISPRole method*), 143
- to_json() (*pymisp.MISPServer method*), 144
- to_json() (*pymisp.MISPShadowAttribute method*), 146
- to_json() (*pymisp.MISPSharingGroup method*), 147
- to_json() (*pymisp.MISPSighting method*), 148
- to_json() (*pymisp.MISPTag method*), 132
- to_json() (*pymisp.MISPTaxonomy method*), 150
- to_json() (*pymisp.MISPUser method*), 133
- to_json() (*pymisp.MISPUserSetting method*), 134
- to_json() (*pymisp.MISPWarninglist method*), 151
- to_json() (*pymisp.tools.ELFObject method*), 162
- to_json() (*pymisp.tools.ELFSectionObject method*), 165
- to_json() (*pymisp.tools.FileObject method*), 160
- to_json() (*pymisp.tools.MachOObject method*), 172
- to_json() (*pymisp.tools.MachOSectionObject method*), 174
- to_json() (*pymisp.tools.PEObject method*), 167
- to_json() (*pymisp.tools.PESectionObject method*), 169
- to_json() (*pymisp.tools.VTReportObject method*), 177
- toggle_global_pythonify() (*pymisp.PyMISP method*), 60, 106
- toggle_warninglist() (*pymisp.PyMISP method*), 61, 106

U

- UnknownMISPObjectTemplate, 68
- unpublish() (*pymisp.MISPEvent method*), 14, 119
- unpublish() (*pymisp.PyMISP method*), 61, 106
- untag() (*pymisp.PyMISP method*), 61, 107
- update() (*pymisp.MISPAAttribute method*), 123
- update() (*pymisp.MISPEvent method*), 119
- update() (*pymisp.MISPEventBlocklist method*), 120
- update() (*pymisp.MISPEventDelegation method*), 121
- update() (*pymisp.MISPFeed method*), 138
- update() (*pymisp.MISPInbox method*), 139
- update() (*pymisp.MISPLog method*), 141
- update() (*pymisp.MISPNoticelist method*), 142
- update() (*pymisp.MISPObject method*), 126
- update() (*pymisp.MISPObjectAttribute method*), 128
- update() (*pymisp.MISPObjectReference method*), 129
- update() (*pymisp.MISPObjectTemplate method*), 130
- update() (*pymisp.MISPOrganisation method*), 136
- update() (*pymisp.MISPOrganisationBlocklist method*), 137
- update() (*pymisp.MISPRole method*), 143
- update() (*pymisp.MISPServer method*), 145
- update() (*pymisp.MISPShadowAttribute method*), 146
- update() (*pymisp.MISPSharingGroup method*), 147
- update() (*pymisp.MISPSighting method*), 149
- update() (*pymisp.MISPTag method*), 132
- update() (*pymisp.MISPTaxonomy method*), 150
- update() (*pymisp.MISPUser method*), 133
- update() (*pymisp.MISPUserSetting method*), 134
- update() (*pymisp.MISPWarninglist method*), 151
- update() (*pymisp.tools.ELFObject method*), 162
- update() (*pymisp.tools.ELFSectionObject method*), 165
- update() (*pymisp.tools.FileObject method*), 160
- update() (*pymisp.tools.MachOObject method*), 172
- update() (*pymisp.tools.MachOSectionObject method*), 174
- update() (*pymisp.tools.PEObject method*), 167
- update() (*pymisp.tools.PESectionObject method*), 169
- update() (*pymisp.tools.VTReportObject method*), 177
- update_analyst_data() (*pymisp.PyMISP method*), 61, 107
- update_attribute() (*pymisp.PyMISP method*), 61, 107

- update_attribute_proposal() (*pymisp.PyMISP method*), 62, 107
 update_decaying_models() (*pymisp.PyMISP method*), 62, 107
 update_event() (*pymisp.PyMISP method*), 62, 107
 update_event_blocklist() (*pymisp.PyMISP method*), 62, 108
 update_event_report() (*pymisp.PyMISP method*), 62, 108
 update_feed() (*pymisp.PyMISP method*), 63, 108
 update_galaxies() (*pymisp.PyMISP method*), 63, 108
 update_galaxy_cluster() (*pymisp.PyMISP method*), 63, 108
 update_galaxy_cluster_relation() (*pymisp.PyMISP method*), 63, 109
 update_misp() (*pymisp.PyMISP method*), 63, 109
 update_not_jsonable() (*pymisp.AbstractMISP method*), 9, 114
 update_not_jsonable() (*pymisp.MISPAttribute method*), 124
 update_not_jsonable() (*pymisp.MISPEvent method*), 119
 update_not_jsonable() (*pymisp.MISPEventBlocklist method*), 120
 update_not_jsonable() (*pymisp.MISPEventDelegation method*), 121
 update_not_jsonable() (*pymisp.MISPFeed method*), 138
 update_not_jsonable() (*pymisp.MISPInbox method*), 139
 update_not_jsonable() (*pymisp.MISPLog method*), 141
 update_not_jsonable() (*pymisp.MISPNoticelist method*), 142
 update_not_jsonable() (*pymisp.MISPObject method*), 126
 update_not_jsonable() (*pymisp.MISPObjectAttribute method*), 128
 update_not_jsonable() (*pymisp.MISPObjectReference method*), 129
 update_not_jsonable() (*pymisp.MISPObjectTemplate method*), 130
 update_not_jsonable() (*pymisp.MISPOrganisation method*), 136
 update_not_jsonable() (*pymisp.MISPOrganisationBlocklist method*), 137
 update_not_jsonable() (*pymisp.MISPRole method*), 143
 update_not_jsonable() (*pymisp.MISPServer method*), 145
 update_not_jsonable() (*pymisp.MISPShadowAttribute method*), 146
 update_not_jsonable() (*pymisp.MISPSharingGroup method*), 147
 update_not_jsonable() (*pymisp.MISPSighting method*), 149
 update_not_jsonable() (*pymisp.MISPTag method*), 132
 update_not_jsonable() (*pymisp.MISPTaxonomy method*), 150
 update_not_jsonable() (*pymisp.MISPUser method*), 133
 update_not_jsonable() (*pymisp.MISPUserSetting method*), 134
 update_not_jsonable() (*pymisp.MISPWarninglist method*), 151
 update_not_jsonable() (*pymisp.tools.ELFObject method*), 162
 update_not_jsonable() (*pymisp.tools.ELFSectionObject method*), 165
 update_not_jsonable() (*pymisp.tools.FileObject method*), 160
 update_not_jsonable() (*pymisp.tools.MachObject method*), 172
 update_not_jsonable() (*pymisp.tools.MachOSectionObject method*), 174
 update_not_jsonable() (*pymisp.tools.PEObject method*), 167
 update_not_jsonable() (*pymisp.tools.PESectionObject method*), 169
 update_not_jsonable() (*pymisp.tools.VTRReportObject method*), 177
 update_note() (*pymisp.PyMISP method*), 63, 109
 update_noticelists() (*pymisp.PyMISP method*), 64, 109
 update_object() (*pymisp.PyMISP method*), 64, 109
 update_object_templates() (*pymisp.PyMISP method*), 64, 109
 update_opinion() (*pymisp.PyMISP method*), 64, 110
 update_organisation() (*pymisp.PyMISP method*), 64, 110
 update_organisation_blocklist() (*pymisp.PyMISP method*), 64, 110
 update_relationship() (*pymisp.PyMISP method*), 65, 110
 update_role() (*pymisp.PyMISP method*), 65, 110
 update_server() (*pymisp.PyMISP method*), 65, 111
 update_sharing_group() (*pymisp.PyMISP method*), 65, 111
 update_tag() (*pymisp.PyMISP method*), 65, 111

update_taxonomies() (*pymisp.PyMISP method*), 66, 111
 update_user() (*pymisp.PyMISP method*), 66, 111
 update_warninglists() (*pymisp.PyMISP method*), 66, 112
 upload_stix() (*pymisp.PyMISP method*), 66, 112
 user_registrations() (*pymisp.PyMISP method*), 66, 112
 user_settings() (*pymisp.PyMISP method*), 67, 112
 users() (*pymisp.PyMISP method*), 67, 112
 users_heartbeat() (*pymisp.PyMISP method*), 67, 113
 users_statistics() (*pymisp.PyMISP method*), 67, 113

V

validate_attribute() (*in module pymisp.tools*), 156
 validate_attributes() (*in module pymisp.tools*), 156
 validate_event() (*in module pymisp.tools*), 156
 validate_object() (*in module pymisp.tools*), 157
 validate_objects() (*in module pymisp.tools*), 157
 ValidationError, 156
 values() (*pymisp.MISPAttribute method*), 124
 values() (*pymisp.MISPEvent method*), 119
 values() (*pymisp.MISPEventBlocklist method*), 120
 values() (*pymisp.MISPEventDelegation method*), 122
 values() (*pymisp.MISPFeed method*), 138
 values() (*pymisp.MISPInbox method*), 140
 values() (*pymisp.MISPLog method*), 141
 values() (*pymisp.MISPNoticelist method*), 142
 values() (*pymisp.MISPObject method*), 126
 values() (*pymisp.MISPObjectAttribute method*), 128
 values() (*pymisp.MISPObjectReference method*), 129
 values() (*pymisp.MISPObjectTemplate method*), 131
 values() (*pymisp.MISPOrganisation method*), 136
 values() (*pymisp.MISPOrganisationBlocklist method*), 137
 values() (*pymisp.MISPRole method*), 143
 values() (*pymisp.MISPServer method*), 145
 values() (*pymisp.MISPShadowAttribute method*), 146
 values() (*pymisp.MISPSharingGroup method*), 147
 values() (*pymisp.MISPSighting method*), 149
 values() (*pymisp.MISPTag method*), 132
 values() (*pymisp.MISPTaxonomy method*), 150
 values() (*pymisp.MISPUser method*), 133
 values() (*pymisp.MISPUserSetting method*), 134
 values() (*pymisp.MISPWarninglist method*), 151
 values() (*pymisp.tools.ELFObject method*), 162
 values() (*pymisp.tools.ELFSectionObject method*), 165
 values() (*pymisp.tools.FileObject method*), 160
 values() (*pymisp.tools.MachObject method*), 172
 values() (*pymisp.tools.MachOSectionObject method*), 174
 values() (*pymisp.tools.PEObject method*), 167
 values() (*pymisp.tools.PESectionObject method*), 170

values() (*pymisp.tools.VTReportObject method*), 177
 values_in_warninglist() (*pymisp.PyMISP method*), 67, 113
 VehicleObject (*class in pymisp.tools*), 156
 version (*pymisp.PyMISP property*), 67, 113
 VTReportObject (*class in pymisp.tools*), 155, 174

W

warninglists() (*pymisp.PyMISP method*), 67, 113